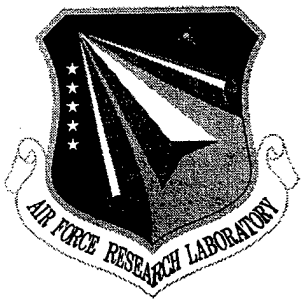
AFRL-IF-RS-TR-1999-187
Final Technical Report
December 1999



OPTOELECTRONIC CACHE MEMORY SYSTEM ARCHITECTURE

University of Pittsburgh

Donald M. Chiarulli and Steven P. Levitan

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1999-187 has been reviewed and is approved for publication.

APPROVED: Bernard J Clarke

Bernard J. Clarke
Project Engineer

FOR THE DIRECTOR: John V McNamara

John V. McNamara
Technical Advisor
Information & Intelligence Exploitation Division

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFED,32 Brooks Road, Rome, NY 13441-4114. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	DECEMBER 1999	Final Sep 96 - Apr 99

4. TITLE AND SUBTITLE
OPTOELECTRONIC CACHE MEMORY SYSTEM ARCHITECTURE

5. FUNDING NUMBERS
C - F30602-96-C-0206
PE - 62702F
PR - 4594
TA - 15
WU - M3

6. AUTHOR(S)
Donald M. Chiarulli and Steven P. Levitan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Pittsburgh
Computer Science Department
212 M1B
Pittsburgh PA 15260

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFED
32 Brooks Road
Rome NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-1999-187

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: Bernard J. Clarke/IFED/(315) 330-2106

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Under this contract, we have designed and implemented an optoelectronic cache memory and interface to a page oriented optical memory system. This work demonstrated that an optical memory can be tightly integrated into a conventional computer and that its high spatial bandwidth can be successfully exploited when optical memories are used as a backing store. Programs, executing on a host connected to this type optoelectronic interface, can transparently access data from the optical store with average access time faster than disk based electronic storage systems. This technology enables the use of large page oriented optical memories in applications such as medical, image, and geo-spatial databases where high speed access to page structured data is essential.

14. SUBJECT TERMS
Optoelectronics, Cache Memory, Volumetric Data Storage, FPGA

15. NUMBER OF PAGES
48

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

TABLE OF FIGURES

1 Technical Overview

1.1 Interface

Hierarchical memory systems have been an integral part of computer system design for nearly three decades. In modern personal computer systems, the virtual address space of a process is implemented on a swapping disk that is at the bottom of a hierarchy consisting of a primary memory and multiple cache levels. Under this contract we have investigated an alternative to this design that implements an optoelectronic (OE) memory hierarchy in which the virtual address space of a process is implemented in an optical page oriented memory. The primary advantage of this structure is the reduction in average memory access time enabled by the parallelism that is inherent in free space transfers of optical memory pages.

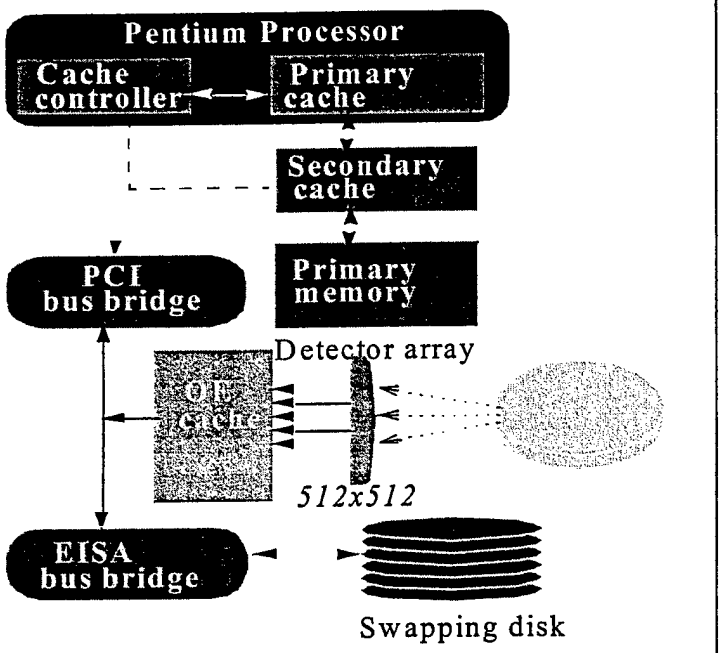


Figure 1: OE Cache Memory System Configuration

A central focus of this effort was an implementation of a proof-of-concept prototype. Figure 1 shows a block diagram of the prototype configuration and Figure 2 is a bitmap image of the OE cache board that was built. The cache board is implemented as an add-on memory controller on the PCI bus of a Gateway 2000 Model P5-133 personal computer. All of the components shown in Figure 1, except for the OE cache, detector array and the optical memory, are packaged in the standard system configuration for this PC. The optical memory is accessed in 512x512 bit pages which are transferred by row through a 512 bit ribbon cable bundle to the OE cache board. The OE cache and controller board is implemented on a long geometry PCI bus add-on card.

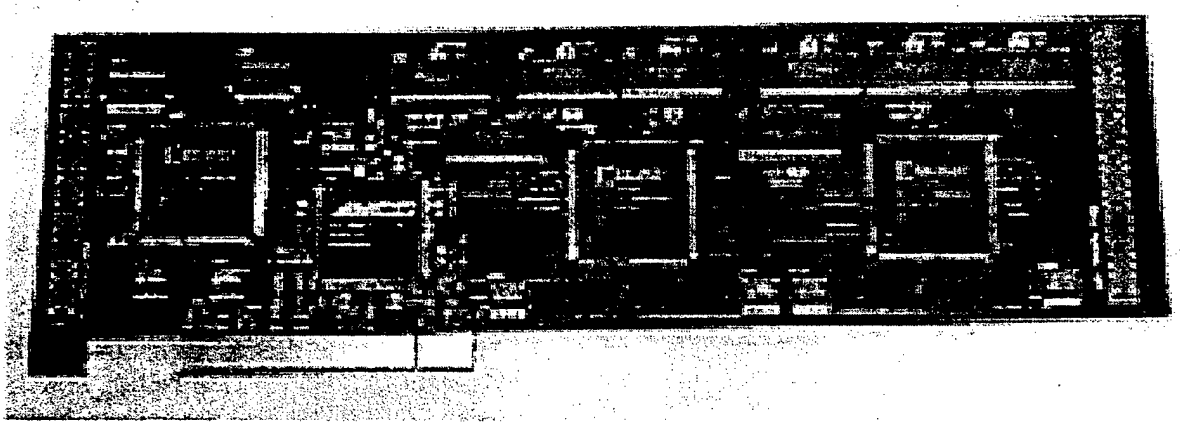


FIGURE 2 – Image of Prototype OE Cache Module

As with its electronic counterpart, the OE memory hierarchy has at its top two levels a primary and secondary cache. The primary cache and the cache controller are integrated into the processor chip. Below these levels is the OE cache that, unlike the primary memory, is logically transparent to the processor address space. This allows the optical memory to appear to the processor as an extension of (or replacement for) the primary memory. Thus, the instructions and data of executing programs are accessed directly from the optical memory with latency hidden by the OE cache.

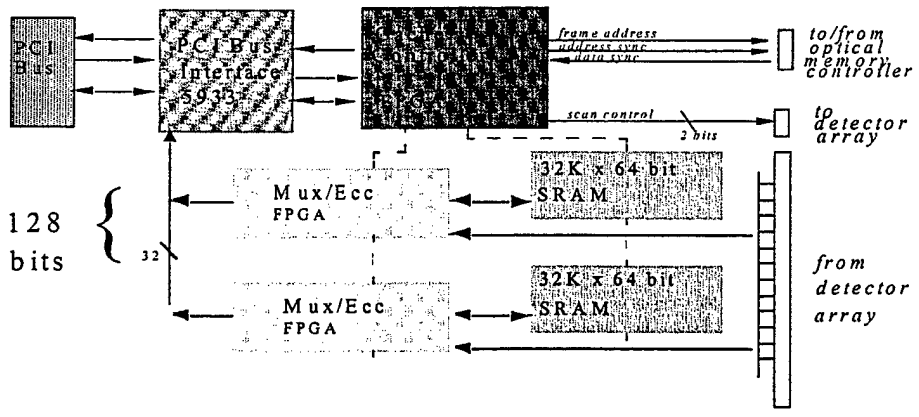


Figure 3: OE Cache Module Block Diagram

Figure 3 shows a block diagram of the OE cache and controller board. The design is based on a "slice" architecture that can be easily adapted to various optical memory page sizes. Two, 128-bit, slices are shown in the figure. Access to the PCI bus is via an AMCC 5933 PCI controller chip that is configured to appear on the PCI bus as an add-on memory controller. It operates in pass-though mode and primarily functions to demultiplex address and data information as well as to provide synchronization between the bus and the optical cache controller. The OE cache controller is implemented in a Xilinx 4K series FPGA. It accepts address and synchronization signals from the PCI controller and uses 4-way set associative mapping to determine if an address is available in the optical cache. If it is, the SRAM modules in the cache are accessed. If not, the address of the requested page is passed on to the optical memory controller. Once the optical memory has been accessed, incoming pages are transferred in parallel by line from the detector array into the OE cache RAMS.

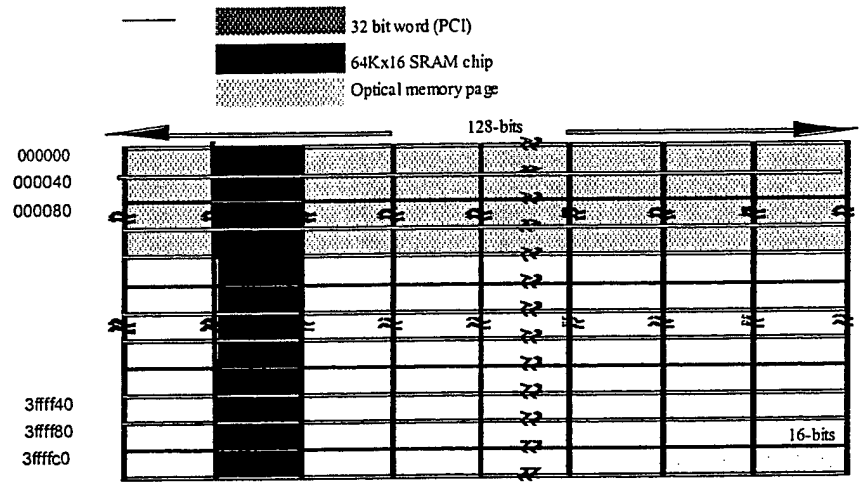


Figure 4: Cache Memory Layout

This implementation requires that the SRAMs in the OE cache are accessible either as 32-bit words on the PCI bus or as 128-bit rows from the optical memory page. Figure 4 shows how this organization is mapped into the memory array by the Mux/Ecc FPGA modules. Each cell in the figure is a 16-bit word and the entire array appears to the OE cache controller as a 4Mb linear address space. These cache addresses are shown to the left of the diagram. Each PCI bus reference accesses up to 4 bytes or memory, such as in the block shown diagonally shaded in the lower right of the figure. The dark shaded column in the figure represents all of the data in a single 64Kx16 SRAM chip. Eight such SRAM chips are accessed in parallel from the optical memory interface and each page of optical memory is loaded into 128 sequential row addresses

as shown by the medium shaded area. Thus, each page of optical memory is mapped into the cache address space as a 32Kb line of OE cache shown as the medium shaded area above.

1.2 Detector

In addition to the interface card we have designed and implemented a 128x128 bit parallel detector array chip for data readout from the optical memory. The readout is an *active pixel sensor* array built under a subcontract to **Parallel Solutions Inc**, of San Diego. The chip receives up to 128x128 optical data inputs on the APS array and two additional dedicated optical signals (these signals can also be received electrically if not available optically, however they would have to be synchronized to the optical data frames). Of these two additional signals, the "optical reset" clears all the on-board registers and latches and the "optical clock" has a half period equal to the period at which images are being imaged to the chip.

A block diagram of the detector chip is shown in Figure 5. A complete description of this device including modes of operations, software interface, and electrical signal definitions can be found in the Interface Control Document included in the appendix.

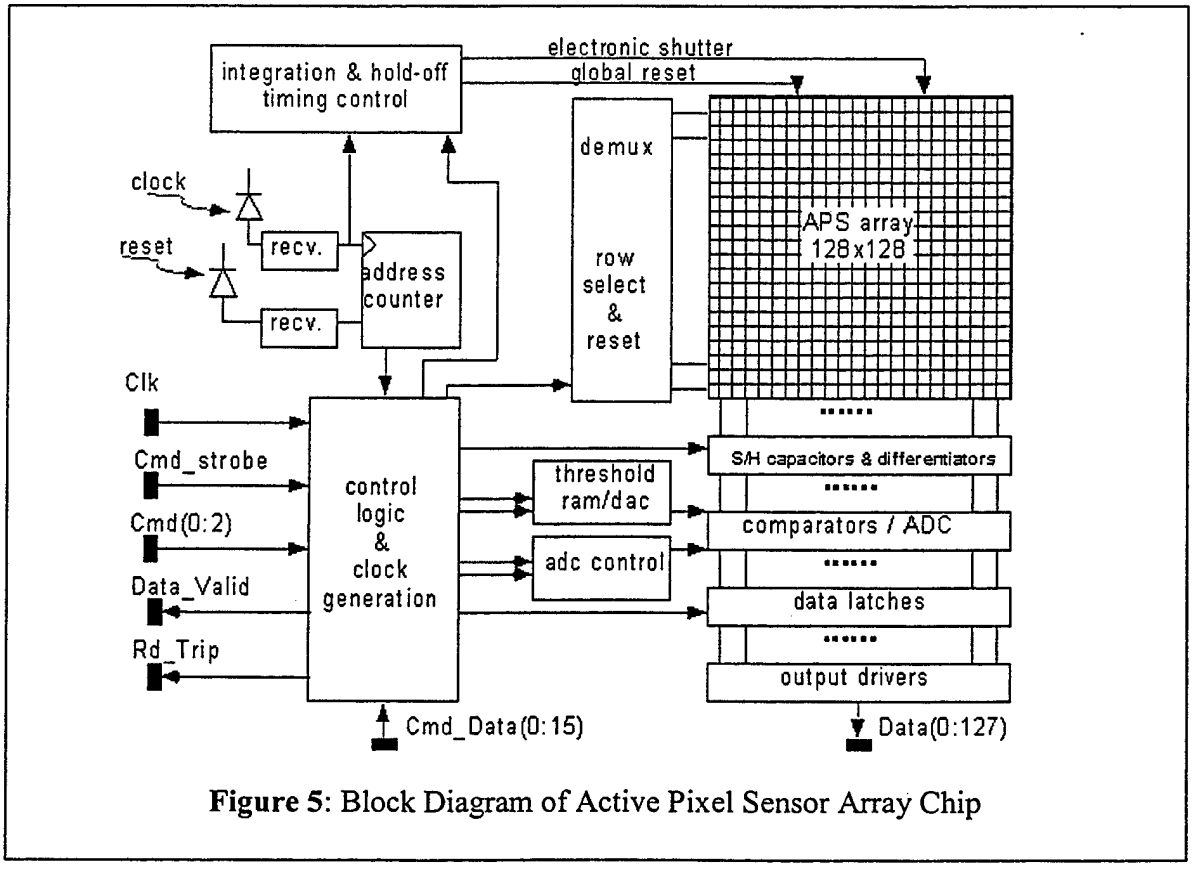


Figure 5: Block Diagram of Active Pixel Sensor Array Chip

1.3 Advance Issues: ECC

In addition to implementation of the interface card and detector array, we have investigated a number of advanced technological issues including the implementation of ECC firmware for the optical memory system. The ECC code used is a 128-bit spectral Reed Solomon Code implemented in Xilinx 4K series FPGAs which are labeled as the Mux/Ecc chips in Figure 3.

The role of the these chips is two-fold. First, they support the data paths necessary for alternately accessing the memory by optical page line and PCI word. Data paths can be established in each chip between 64 bits of an incoming optical memory line and the SRAM I/O line. In another configuration any 16 of the 64 SRAM I/O lines can be selected for transfer to the PCI bus. Second these chips provide for the introduction of Error Checking and Correcting (ECC) logic operating over the incoming data stream. Error correction is performed in real time on each line of the incoming data. The OE cache controller handles synchronization and controls the transfer. While loading the incoming optical data, the cache controller simultaneously monitors the data stream and latches the requested word into an internal buffer. This allows the memory request to be satisfied without the additional cycles required for a second access to the OE cache SRAMS.

1.3.1 Reed Solomon Codes

Reed-Solomon codes are a class of non-binary, linear block codes that offer multiple error correcting capability. Unlike a binary code, a linear block code treats the data to be encoded as message block of k symbols where each symbol is encoded in m-bits and represents an element of a finite field of size 2^m . An example codeword is shown in Figure 6. The codeword is built by appending $(n\text{-}k)$ extra symbols to the message string to produce a block with a total length of n symbols. This nxm-bit word is referred to as an (n,k) Reed-Solomon (RS) code and has random error correction capability $t = (n\text{-}k)/2$ symbols. In the FPGA implementation a sixteen element field was chosen yielding m=4, n=15 and k=9 and a 60-bit, (15,9) RS code word.

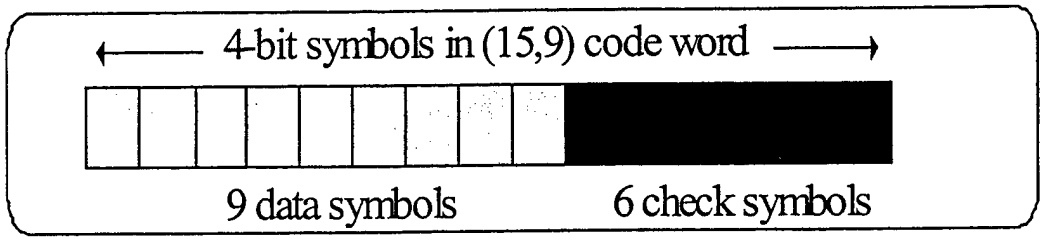


Figure 6: Sample Codeword

The specifics of the encoding and decoding operations are shown in Figure 7. An n-symbol unencoded data word is assumed to consist of k data symbols and n-k check symbols. The n-k check symbols are initially set to the zero symbol, the additive identity element in the field. For coding purposes, the entire string of symbols is interpreted as a vector, V, in the spectral domain. The encoding operation is an inverse Finite Field Fourier Transform (FFFT) which produces a vector of symbols, v, in the temporal domain. This is the encoded data word that is stored in the memory system.

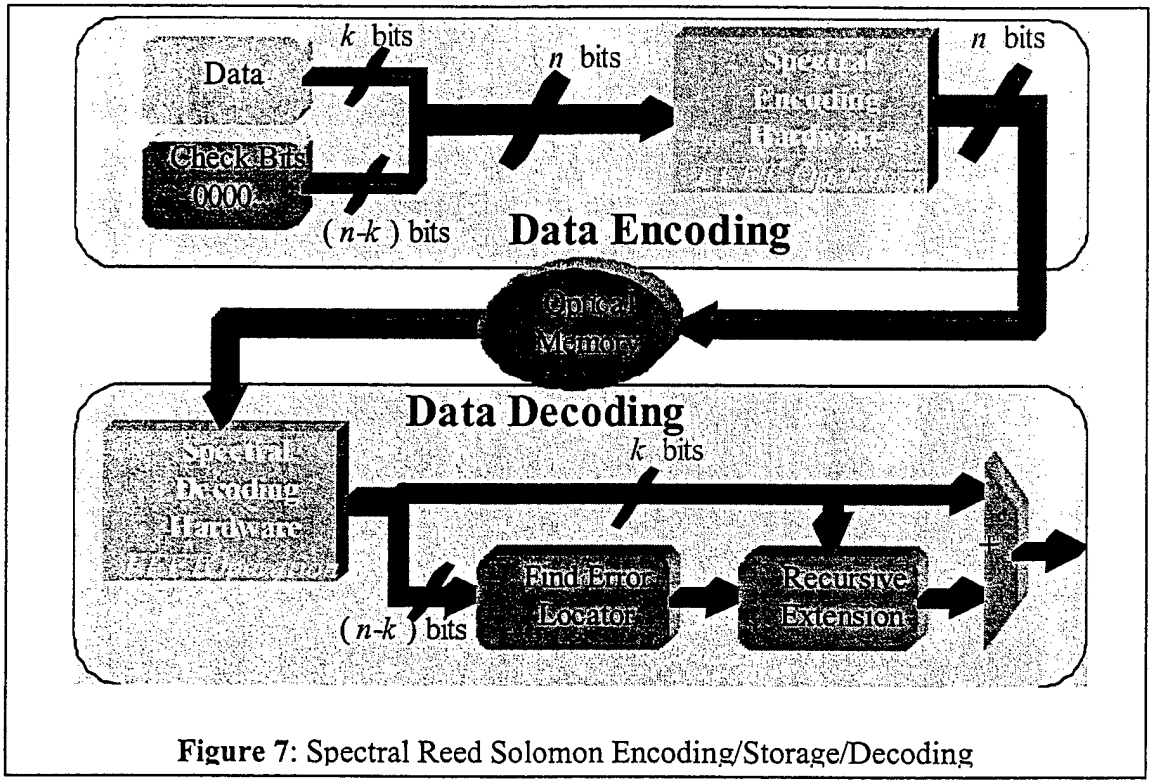


Figure 7: Spectral Reed Solomon Encoding/Storage/Decoding

After being stored and retrieved from the optical memory, the data is processed by the multi-stage decoding and correction logic. The decoding step consists of a forward FFFT to restore the original spectral data vector. If errors were introduced during storage and retrieval, the error can be represented in temporal space as a vector e such that the received vector v is the (finite field) sum of code word d and error vector e.

$$v = d + e$$

After decoding, the linearity property of the FFFT operation provides that:

$$V = D + E$$

also holds in the spectral domain. However, since six of the D terms in the original code word were appended as zero constants, the symbols in this portion of the code word represent elements in the error vector only. These non-zero elements in the check portion of the retrieved vector V

are the starting point for the correction algorithm. The goal of the correction algorithm is to calculate the remaining elements in the error vector E, such that when these elements are added to the elements of V, the original data is restored.

The first stage of error correction calculates the coefficients of a function called the error locator polynomial. The error locator polynomial, $\Lambda(x)$, is defined such that if an error, e_i, occurs at location i in the temporal space then α^{-i} is one of its roots. Due to the way in which $\Lambda(x)$ is defined, the inverse FFFT of this polynomial, $\lambda(x) = \text{FFFT}^{-1}(\Lambda(x))$, has the very important property that $\lambda_i * e_i = 0, \quad \forall \quad i \in \{0 \ldots n-1\}$. Application of the convolution theorem results in this set of n equations, which can be used to determine the coefficients of Λ and the unknown components of E:

$$\sum_{j=0}^{t} \Lambda_j E_{i-j} (\text{mod}\, x^n) = 0, i = 0,1,\ldots n-1$$

There will exist a subset of t of these equations which contain only the 2*t known components of E and the t + 1 coefficients of Λ. Making use of the fact that $\Lambda_0 = 1$, this reduces to a system of t equations with t unknowns which can be solved for the values of $\Lambda_1, \ldots, \Lambda_t$. This solution can be obtained through several different methods, our choice being the iterative Berlekamp-Massey (BM) algorithm.

Once the error locator polynomial has been calculated, the Λ coefficients can be used with the remaining equations from the convolution result to calculate the unknown symbols of E in a process known as recursive extension (RE). The RE stage determines the remaining terms of the error vector by stepwise solution of the remaining equations, producing one new term per equation. Finally each of the calculated terms in the error vector is added (equivalent to subtraction in a finite field) to the retrieved data to regenerate the original data.

We have verified the functional correctness of the algorithm using a software implementation. Figure 8 shows a sequence of images beginning with raw unencoded data, followed in sequence by the encoded data, encoded data with errors introduced, decoded and uncorrected data, and corrected data.

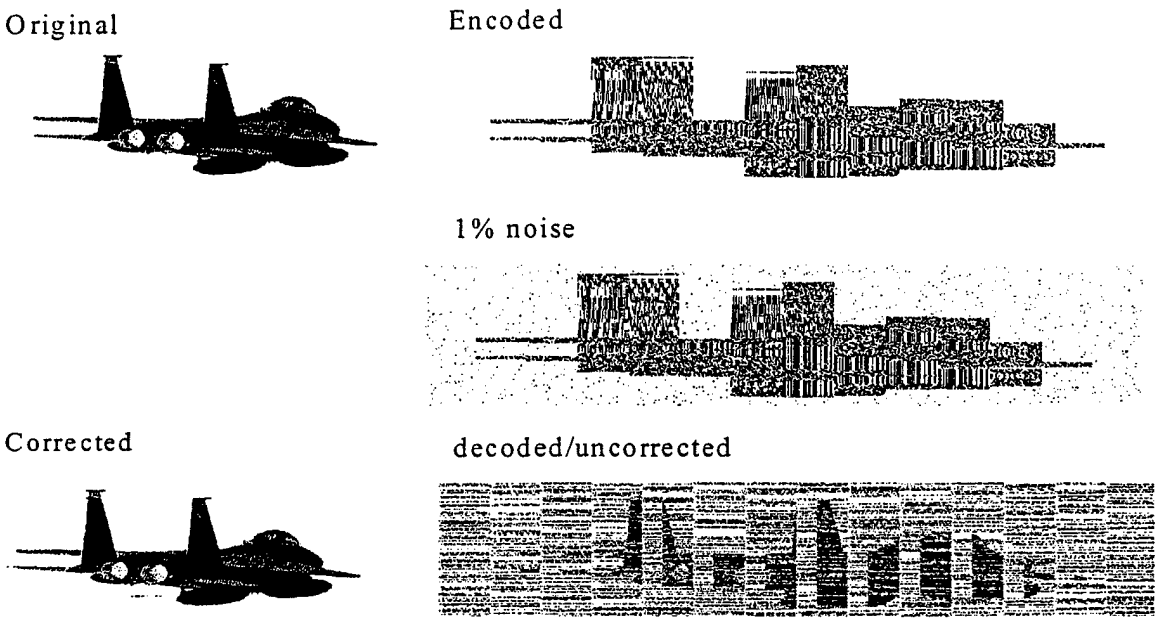


Figure 8 : Functional Simulation of ECC Algorithm

1.3.2 Firmware Architecture

In this section, we describe the implementation of a Spectral RS decoder in a single Xilinx XC4036EX, field programmable gate array (FPGA) device. As we mentioned above, this is a firmware solution that allows different versions of the software to be loaded as necessary for testing, debugging, and optimizing alternate configurations. The circuitry itself is designed in VHDL (*V*HSIC *H*ardware *D*escription *L*anguage), which provides for a textual description of the algorithm much like a conventional programming language. Software tools synthesize the logic directly from this description and place and route tools map the logic into the combinational logic blocks (CLBs) and interconnection network of the FPGA.

One of the principle advantages of this firmware approach is to allow multiple versions of the ECC algorithm to be implemented on the same hardware. Given that the topological error characteristics of various optical memory materials may vary, we have anticipated this fact by implementing several versions of the ECC algorithm that differ in the way that code words are mapped to the surface of a memory page. Figure 8 show four examples which range from a linear mapping of the 15 symbols across of single horizontal line, through various blocking patterns, to a vertical mapping in which each symbol of an input line is from a separate code word. This has a secondary hardware/performance tradeoff effect, that will be explained below after a brief outline of the FFFT implementation.

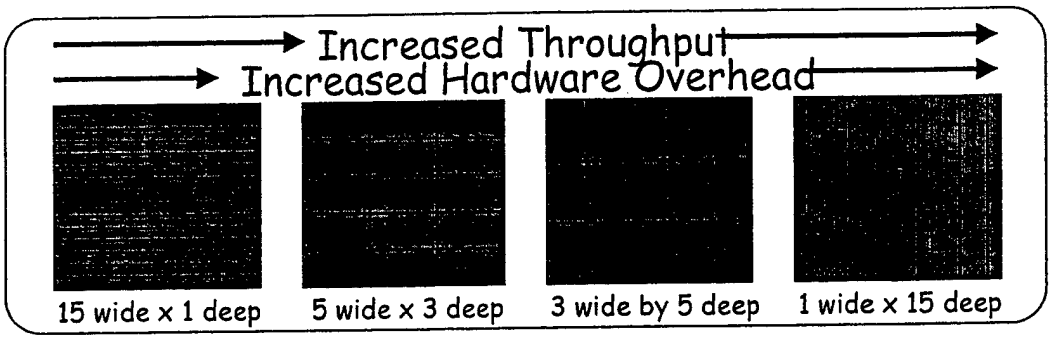


Figure 9: Topological Mapping Options for code words to memory on memory Page

As explained above the *k*-th symbol of the FFFT is computed from:

$$V_k = \sum_{i=0}^{n-1} \alpha^{i*k} v_k$$

where the operations for multiplication and addition are defined by the finite field algebra. In $GF(2^4)$, where each symbol consists of four bits, these operations are defined by a combinational logic function of 8 inputs and 256 outputs. While addition is easily implemented as a bit-wise XOR operation, multiplication cannot be readily minimized and requires over 30 gates, (5 CLBs in an FPGA) to implement. Although the most obvious realization of the FFFT operation is to compute V_i by a multiplication addition tree, this would require some 225 multipliers and would consume most of the capacity of available reconfigurable logic devices. However, unlike a conventional FFT operation, finite field FFT's can be implemented such that one of the two input symbols, α^{i*k}, is known *a-priori* for each of the 15 multiply operations of the *k*-th term. Thus, each of these multiply operations can be implemented as a 4 input, 16 output function which can be implemented in 2 CLBs. Using these simpler multiply operations, the multiply-add circuitry for the *k*-th term can be built as shown in Figure 9.

To complete the implementation of the FFFT, pipeline registers must be added in order to meet timing specifications. These are arranged in two groups, one group along the rows of the FFFT array such that a new partial sum for each term in V is computed in each clock cycle.

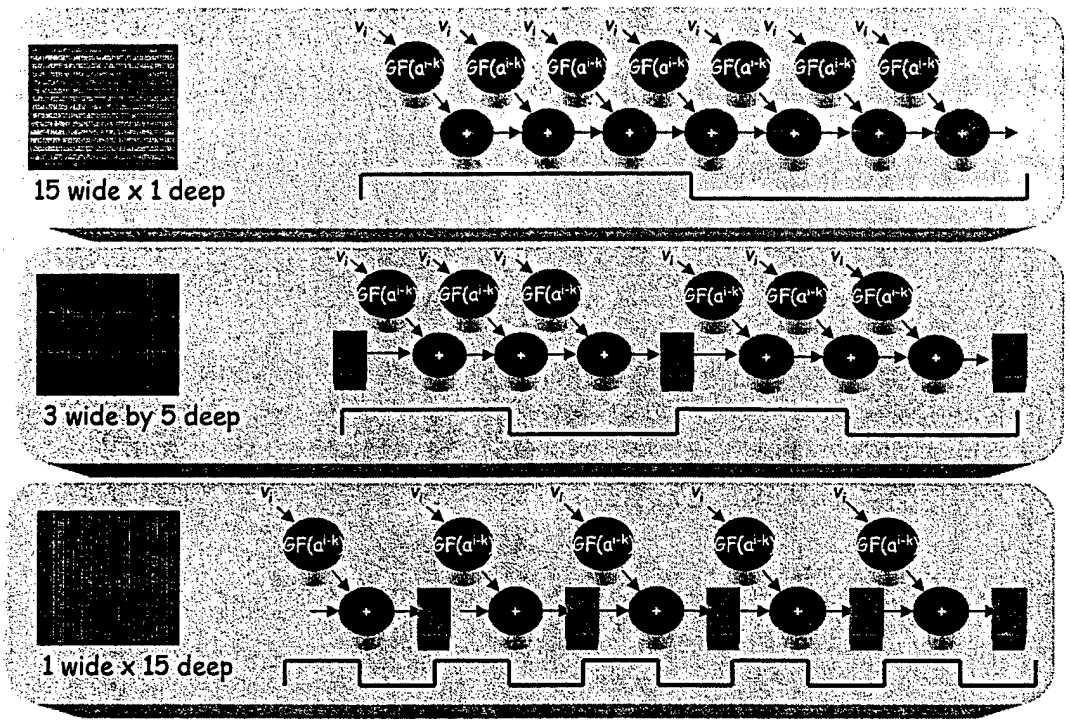


Figure 10: FFFT Pipeline organization for several codeword topologies

These registers are shown as green blocks in Figure 10. Another group of registers, not shown in Figure 10, buffers the input code word to delay the input of each v term until the partial sum has propagated to the corresponding column. Note the difference in pipeline depth and clock rate based in the code word topology. All three of the configurations shown generate one complete FFFT per clock cycle. However, the deepest pipeline has a 15 times greater throughput rate since it computes one partial result for each of 15 different code words on each clock cycle. The cost of this speedup is additional hardware for pipeline registers and buffering.

Figure 11 shows measured performance data for several implementations with varying pipeline depth. For one, three, and five-deep pipelines, performance improves as expected with pipeline depth. However, an interesting anomaly occurs for the deepest pipeline. Since these designs are being synthesized in Xilinx 4K series FPGAs, device clock rates are limited by the number of CLBs that lie along the critical delay path. Since each CLB contains part of the logic and two bits of the pipeline register for each stage, the number of CLBs in a path depends on the ratios of logic within each pipeline stage to the number of bits in the pipeline register. The 15 deep pipeline exceeds the optimal ratio and the design must be spread across multiple CLBs in order to support additional pipeline register bits.

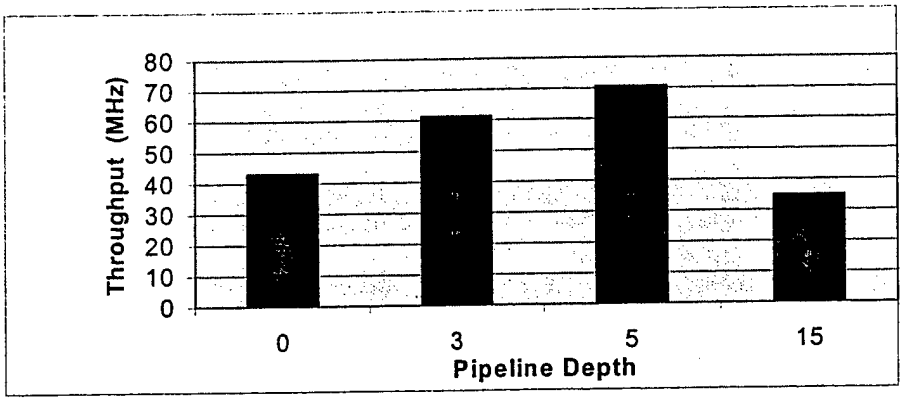


Figure 11: FFFT Pipeline Clock Rate (Throughput) vs Depth

The final stages of the ECC implementation, the Berlekamp-Massey(BM) computation and it's recursive extension are described in a paper included in the appendix to this report.

2 Specific Accomplishments

The following tasks were accomplished. Details of the design including system organization, performance data, multi-threshold detector interface, and firmware algorithms for the cache controller and the pipelined SRS/ECC algorithms were published in numerous technical publications. Copies of these publications are attached in the Appendices.

1. Design and fabrication of a prototype optoelectronic cache memory interface board including:
   - Design of a PCI Bus compliant 8 level printed circuit board
   - Mixed 5 and 3.3 Volt supplies for minimum power consumption
   - Support for 16MB of SRAM cache
   - 128 x 128 bit optical memory page size expandable to 512 x 512
   - High speed, row parallel loading at 33MHz
   - PCI interface
   - Controller and interface for detector chip and optical memory
   - Cache memory/optical detector interface in FPGA firmware
   - Full caching supported using NUR/LRU algorithm
   - Implementation of a new Linux O/S kernel to enable the interface to work transparently with application programs
   - Development of a Multiple Sample/Threshold detector interface

2. Design of a 128x128 APS detector array
   - 128x128 Input pixels
   - 35x35 μm^2 pixels with 45% fill factor (could reach 100% if micro-lenses were integrated on top of the chip)
   - Maximum Optical Frame Rate: 200 KHz
   - Minimum Optical Power: 1 nW/pixel
   - Minimum Contrast Required at minimum power level: 2:1
   - Max. Digital Output: 128 lines at 50 Mbits/sec ==> 6.4 Gbits/sec
   - Chip Area: 8x8 mm^2
   - 128 on-chip parallel 8-bit A/D: 800 Msamples/sec

- Comparison to 16 programmable thresholds per pixel
- Programmable Array Size (within 128x128)
- Programmable Integration Time
- Single frame or continuous readout

3. Implementation of Spectral Reed Solomon ECC Firmware
   - Spectral Reed-Solomon ECC coding research and development for page oriented optical memory applications
   - Implementation of highly pipelined SRS/ECC codes in FPGA
   - Implementation of Cache Controller, Bus Multiplexer, ECC in 50MHz FPGA technology

The results of this effort are embodied in the final Programmable O/E Cache interface board completed at the University of Pittsburgh in September of 1998 and delivered to the Air Force Research Laboratory.

3 Student Support

Graduate Students:

Robert Hofmann (Physics)

Leo Selavo(C.S.)

Madhulima Pandey (M.S.E.E.)

Timothy Kurzweg (M.S.E.E.)

Jose Martinez (E.E.)

Undergraduate Students:

Paige Derr (BS-Physics)

Debra Davis (CAS)

Danielle Dixon (CAS)

Mark Hilliard (BSEE)

Jason Loomis (BSEE)

Rajon Gilmore (EE)

4. Publications

1. "An Optoelectronic Cache Memory System Architecture," D.M. Chiarulli, S.P. Levitan, *Applied Optics,* Vol. 35, No. 14, pp. 2449-2456, 10 May 1996.

2. "Optoelectronic Cache Memory System Architectures," D.M. Chiarulli, S.P. Levitan, Workshop on Data Encoding for Page Oriented Optical Memories (DEPOM'96), pp. 23-28, Phoenix, AZ, Mar. 1996.

3. "Making Virtual Memory Real: Integrating an Optical Memory into the Memory Hierarchy," D.M. Chiarulli, S.P. Levitan, Workshop on Optics and Computer Science, 1997 International Parallel Processing Symposium (IPPS'97), Geneva, Switzerland, Apr. 1-5, 1997.

4. "Design and Implementation of an Optical Page Oriented Virtual Memory for a Personal Computer," D.M. Chiarulli, S.P. Levitan, OSA Optics in Computing Spring Topical Meeting (OC'97), OThD (poster), pp. 198-200, Incline Village, NV, Mar. 18-21, 1997.

5. "Spectral Reed-Solomon Coding for Optical Memories", Madhulima Pandey, Technical Report, Department of Electrical Engineering, University of Pittsburgh, July 1997.

6. "Error Detection and Correction for an Optoelectronic Memory System," R. Hofmann, M. Pandey, S.P. Levitan, D.M. Chiarulli, Special Session on Optics in Communications and Computing, International Conference on Telecommunications (ICT'98), Porto Carras Resort, Chalkidiki, Greece, June 22-25, 1998.

7. "Error Detection and Correction for an Optoelectronic Cache Memory Interface," D.M. Chiarulli, R. Hofmann, S.P. Levitan, M. Pandey, SPIE Annual Meeting, Vol. 3468-12, San Diego, CA, July 21-24, 1998.

5 Invited Presentations

1. "High Fidelity Interface in an Optoelectronic Memory Hierarchy", AFOSR, Workshop on Materials and Interfaces to Spectral Memory Systems", Bozeman Montana, March 1998
2. "Integrating Optical Storage into Real-World Computing Systems," ARO Asilomar Workshop on Rare Earths for 3-D Optical Devices, Asilomar, CA, Apr. 1998.
3. "Building a better bathtub: Computing at the optical memory interface", D. M. Chiarulli, "Advanced Optical Data Storage: Materials, Systems, and Interfaces to Computers Technical Conference, Denver, CO, July 20-22,1999

APPENDIX A: TECHNICAL PUBLICATIONS

Design and Implementation of an Optical Page Oriented Virtual Memory for a Personal Computer

Donald M. Chiarulli, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, 15260 +1-412-624-8839, FAX: +1-412-624-5249

Steven P. Levitan, Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA, 15260 +1-412-648-9663, FAX: +1-412-624-8003

Summary

We describe a prototype system which transparently places a page oriented optical memory into the memory hierarchy of a personal computer.

Design and Implementation of an Optical Page Oriented Virtual Memory for a Personal Computer

Donald M. Chiarulli
Department of Computer Science
University of Pittsburgh, Pittsburgh PA 15260

Steven P. Levitan
Department of Electrical Engineering
University of Pittsburgh, Pittsburgh PA 15260

Hierarchical memory systems have been an integral part of computer system design for nearly three decades. In modern personal computer systems, the virtual address space of a process is implemented on a swapping disk which is at the bottom of a hierarchy consisting of a primary memory and multiple cache levels. This paper describes an alternative to this design that implements an optoelectronic (OE) memory hierarchy in which the virtual address space of a process is implemented in an optical page oriented memory. The primary advantage of this structure is the reduction in average memory access time enabled by the parallelism which is inherent in free space transfers of optical memory pages.

In previously published results[1] we have presented simulation data which suggests that a reduction of several orders of magnitude in the average memory access time is possible when compared to a disk-based/electronic memory hierarchy. In this discussion we present the design of a prototype system being built at the University of Pittsburgh in collaboration with 2-photon 3D memory group at UCSD[2]. We are also investigating a number of issues relating to address mapping, replacement strategy, and memory coherence in the context of an OE memory hierarchy.

Figure 1 shows a block diagram of the prototype. The prototype is implemented as an add-on memory controller on the PCI bus of a Gateway 2000 Model P5-133 personal computer. All of the components shown in Figure 2, except for the OE cache, detector array and the optical memory, are packaged in the standard system configuration for this PC. The optical memory is accessed in *512x512* bit pages which are transferred by row through a 512 bit ribbon cable bundle to the OE cache board. The OE cache and controller board is implemented on a long geometry PCI bus add-on card.

As with its electronic counterpart, the OE memory hierarchy has at its top two levels a primary and secondary cache. The primary cache and the cache controller are integrated into the processor chip. Below these levels is

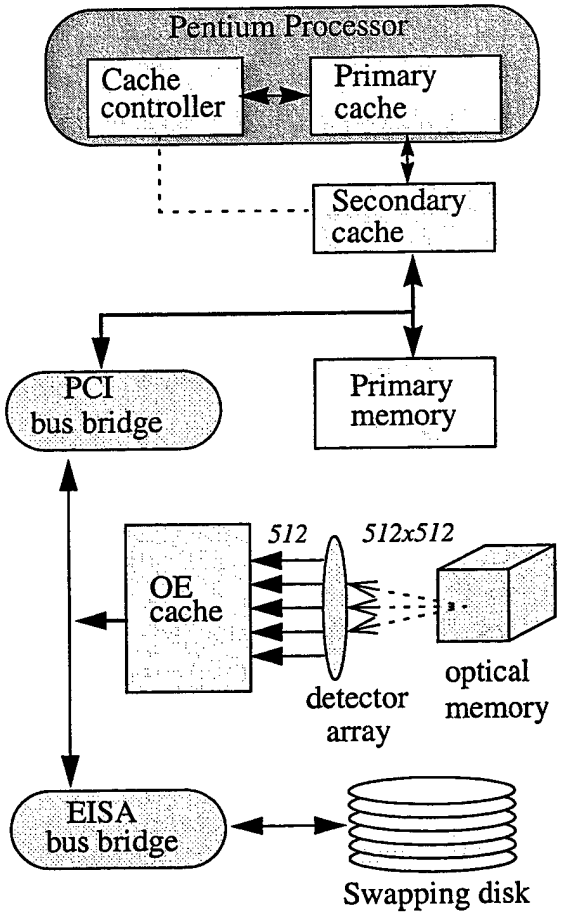


Figure 1: Prototype Block Diagram

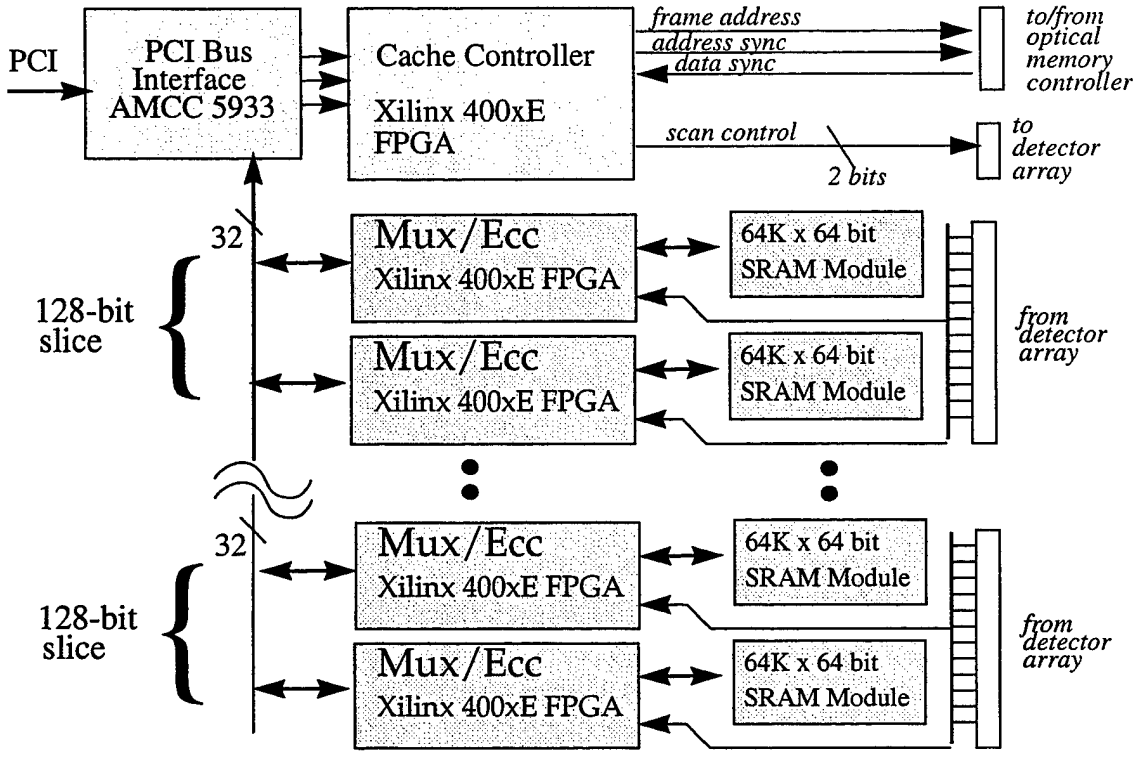


Figure 2: OE Cache Block Diagram

the OE cache which, unlike the primary memory, is logically transparent to the processor address space. This allows the optical memory to appear to the processor as an extension of (or replacement for) the primary memory. Thus, the instructions and data of executing programs are accessed directly from the optical memory with latency hidden by the OE cache.

Figure 2 shows a block diagram of the OE cache and controller board. The design is based on a "slice" architecture which can be easily adapted to various optical memory page sizes. Two, 128-bit, slices are shown in the figure. Access to the PCI bus is via an AMCC 5933 PCI controller chip which is configured to appear on the PCI bus as an add-on memory controller. It operates in pass-though mode and primarily functions to demultiplex address and data information as well as to provide synchronization between the bus and the optical cache controller. The OE cache controller is implemented in a Xilinx 4K series FPGA. It accepts address and synchronization signals from the PCI controller and uses 4-way set associative mapping to determine if an address is available in the optical cache. If it is, the SRAM modules in the cache are accessed. If not, the address of the requested page is passed on to the optical memory controller. Once the optical memory has been accessed, incoming pages are transferred in parallel by line from the detector array into the OE cache RAMS.

This implementation requires that the SRAMs in the OE cache are accessible either as 32-bit words on the PCI bus or as 512-bit rows from the optical memory page. Figure 3 shows how this organization is mapped into the memory array by the Mux/Ecc modules. Each cell in the figure is a 16-bit word and the entire array appears to the OE cache controller as a 4Mb linear address space. These cache addresses are shown to the left of the diagram. Each PCI bus reference accesses up to 4 bytes or memory, such as in the block shown diagonally shaded in the figure. The dark shaded column in the figure represents all of the data in a sin-

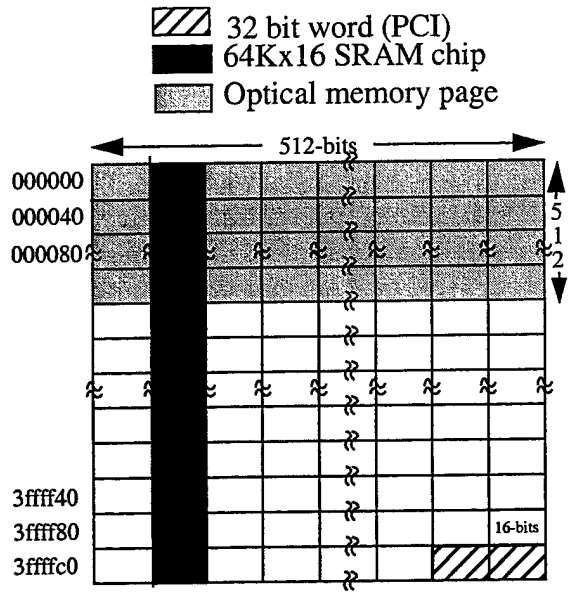


Figure 3: Cache memory addressing

gle 64Kx16 SRAM chip. Thirty two such SRAM chips are accessed in parallel from the optical memory interface and each page of optical memory is loaded into 512 sequential "row" addresses as shown by the medium grey shaded area. Thus, each page of optical memory is mapped into the cache address space as a 32Kb line of OE cache.

One effect of this large cache memory line is the introduction of *internal fragmentation.* This is an effect in which significant portions of incoming block go unused because the size of the block exceeds to locality characteristics of the application. Excessive internal fragmentation can reduce the efficiency of cache utilization and reduce the hit ratio. However, in the OE cache, it is possible to partition the incoming optical page in separate logical pages which can be independently mapped into OE cache lines by adjusting the line addresses of each partition.

Figure 4 shows a diagram of the contents of the Mux/ECC chips. Also implemented in Xilinx 4K series FPGAs, the role of the these chips is two-fold. First, they support the data paths necessary for alternately accessing the memory by optical page line and PCI word. As shown by the dotted lines in the figure, data paths can be established in each chip between 64 bits of an incoming optical memory line and the SRAM I/O line. In another configuration any 16 of the 64 SRAM I/O lines can be selected for transfer to the PCI bus. Second these chips provide for the introduction of Error Checking and Correcting (ECC) logic operating over the incoming data stream. The reconfigurable logic devices as well as the position of the chips relative to the data streams allows experimentation with a variety of algorithms operation on the data either as it is incoming from memory, or outgoing to the PCI bus.

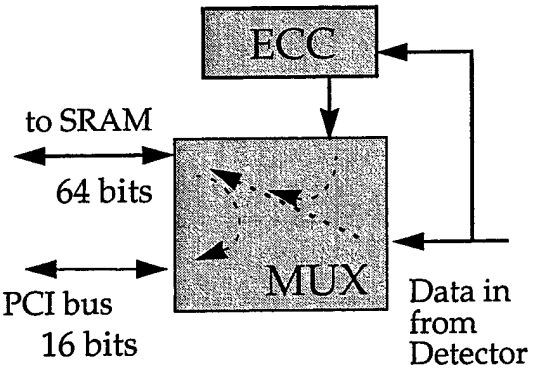


Figure 4: Bus Mux/ECC configurations

Alternative implementations are also under study using smart pixel arrays to combine the functions of the detector array and OE cache RAM onto a single device. This implementation will support the full parallel transfer capability of the optical memory.

This work has been supported in part by the United States Air Force, AFMC/Rome Laboratory/PKRL, under contract #I-6-4284.

References

[1] D.M. Chiarulli and S.P. Levitan, "An Optoelectronic Cache Memory System Architecture", Applied Optics Vol. 35, No. 14,pp. 2449-2456, 10 May 1996

[2] P. A. S. Dvornikov, S. Esener, and P. Rentzepis,"Three-dimensional optical storage by means of photon interactions,"Optical Computing Hardware, ed. Jahn and Lee, Academic Press 1994.

Making Virtual Memory Real: Integrating an Optical Memory Into the Memory Hierarchy

Donald M. Chiarulli, Department of Computer Science
Steven P. Levitan, Department of Electrical Engineering
University of Pittsburgh, Pittsburgh, PA, 15260

Abstract

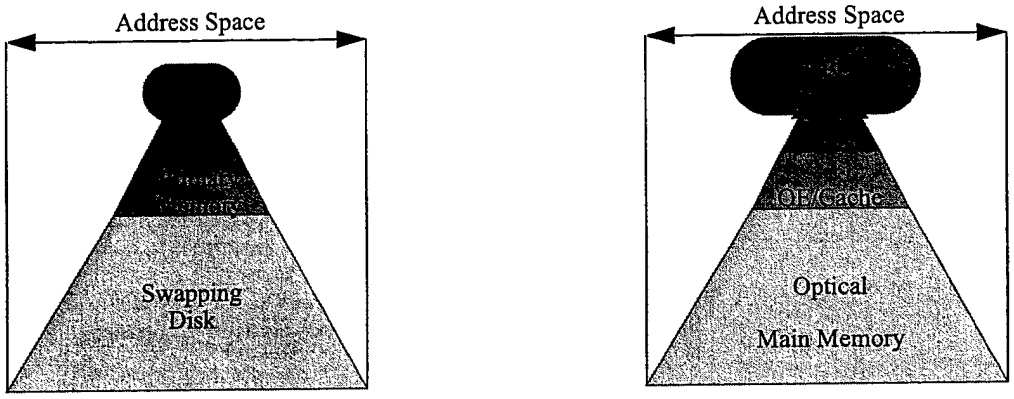
We present the architecture of an optoelectronic cache that integrates an optical memory into the memory hierarchy of a personal computer, thereby supporting terabit address spaces with effective access times comparable to the cycle time of the CPU. This enables optical terabit technologies to transparently provide low latency secondary memory with frame sizes comparable to disk-pages and speed approaching that of electronic secondary cache memories. We describe a prototype system which transparently places a page oriented optical memory into the memory hierarchy of a personal computer and simulation results which predict the performance of this system.

Introduction

One of the most significant trends in computer architecture in recent years has been the continuing increase in the ratio of the access time for large memories to the cycle time of clock level CPU operations. In order to reconcile this difference, system designers have adopted hierarchical memory systems which exploit locality behavior in memory access patterns to hide the latency of the memory system. For example, in modern personal computer systems, the address space of a process is implemented on a swapping disk which is at the bottom of a hierarchy, such as the one shown in Figure 1(a), which includes a primary memory level and multiple cache levels. In such a system, the primary cache level, built on the processor chip, typically captures 95% or more of the memory references and responds to these with an access time of one or two CPU clock cycles. The remaining references must access the larger and slower lower levels of the hierarchy. Each level in turn replaces a block of memory when accessed, rather than a single location in the level above. By replacing a block, the access time of the lower level is amortized over a collection of future references which locality behavior predicts will be confined to a small set of such blocks. Thus, the average memory access time of the system is dominated by the large percentage of primary cache accesses.

This paper describes an alternative implementation based on an optoelectronic (OE) memory hierarchy in which the address space of a process is implemented in an optical page oriented memory. This alternative, as shown in Figure 1(b), retains the primary and secondary cache levels of the electronic hierarchy but replaces the swapping disk with an optical page-oriented memory and the primary memory with a new level, called the OE cache level. Unlike primary memory, the OE cache level is transparent to both the processor and the operating system. The address space of a process is instantiated directly in the optical memory. The OE cache, receives pages of optical memory data in parallel through a free space optical interconnect. Each page is stored in cache and treated as a single block. On the processor side, the same data can be transferred from the OE cache to the secondary cache along the system bus.

A distinguishing feature of the OE cache is its significantly larger *line size* than is typical for primary memory. A *memory line*, (also commonly known as a cache line) is the amount of data transferred between levels of the hierarchy when a memory fault (or equivalently, a cache miss) occurs. Thus, the size of a line at a particular level is a trade-off between the locality supported within the memory traffic and the efficiency to which the cache is utilized. A large cache line more loosely constrains memory access locality. However, large cache lines may also bring into the cache fragments of unused memory. This effect is called *internal fragmentation.* In a conventional memory the cost associated with internal fragmentation can be significant since the fault service time is typically linearly related to the line size. However, in the OE cache, the (much larger) line size is determined by the width of the optical memory word. The parallel access characteristics of an optical memory make it possible to transfer cache



(a) Modern Systems (b) OE Architecture

Figure 1: Hierarchical Memory System

lines to and from the optical memory in a single access time. This is substantially faster than the equivalent transfer from a magnetic disk which must allow for both rotational latency and serial transfers. In previously published results[1] we have presented simulation data which suggests that a reduction of several orders of magnitude in the average memory access time is possible when compared to a disk-based/electronic memory hierarchy. We are also investigating a number of issues relating to address mapping, replacement strategy, and memory coherence in the context of an OE memory hierarchy. In this paper we present the design of a prototype system being built at the University of Pittsburgh in collaboration with 2-photon 3D memory group at UCSD[2].

Prototype system

Figure 2 shows a block diagram of a prototype personal computer system with an optoelectronic memory system. The prototype is implemented as an add-on memory controller on the PCI bus of a Gateway 2000 Model P5-133 personal computer. All of the components shown in Figure 2, except for the OE cache, detector array and the optical memory, are packaged in the standard system configuration for this PC. The optical memory is accessed in *512x512* bit pages which are captured on an external detector head and are transferred by row through a 512 bit ribbon cable bundle to the OE cache board. A single, long geometry PCI bus board implements the OE cache memory, bus multiplexing circuits, and the cache controller.

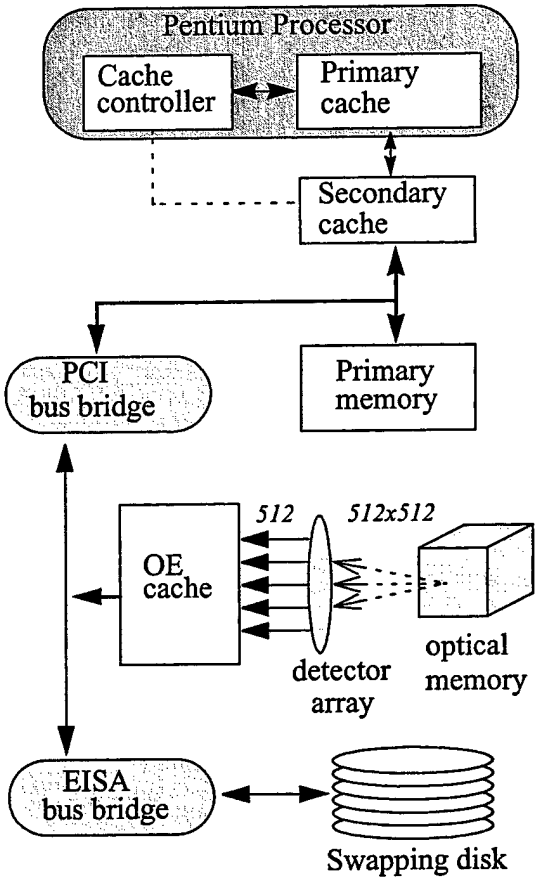


Figure 2: Prototype Block Diagram

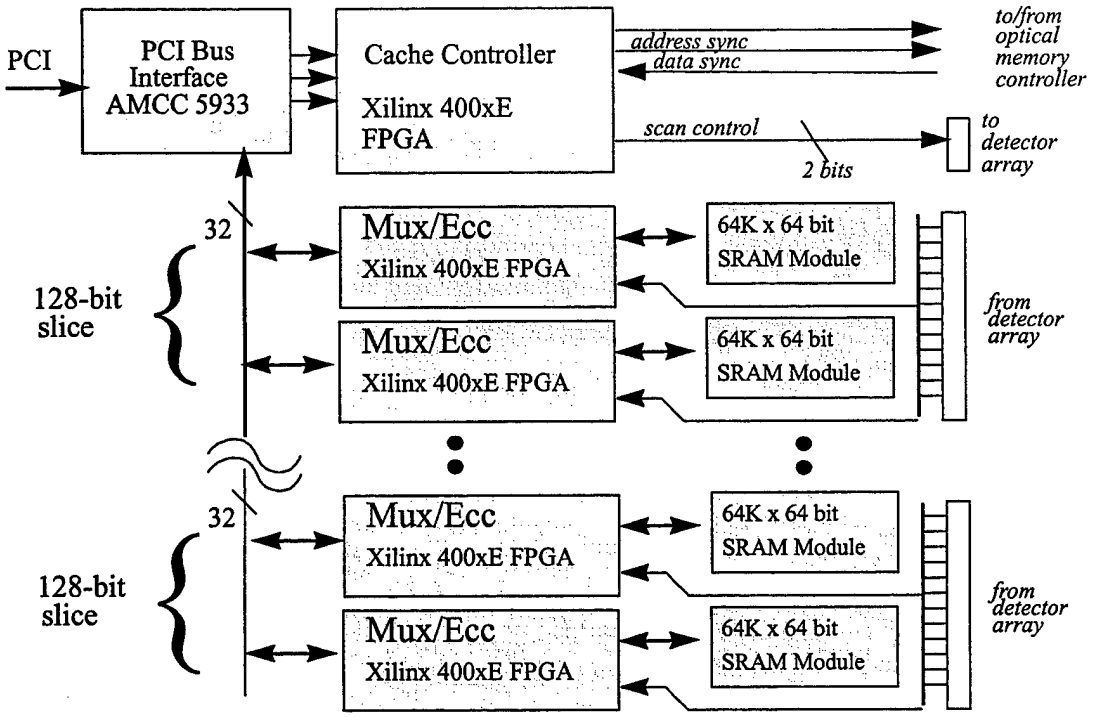


Figure 3: OE Cache Controller Block Diagram

Figure 3 shows a block diagram of the OE cache and controller board. The design is based on a "slice" architecture which can be easily adapted to various optical memory page sizes. Two, 128-bit, slices are shown in the figure. Access to the PCI bus is via an AMCC 5933 PCI controller chip which operates in pass-though mode and primarily functions to demultiplex address and data information as well as to provide synchronization between the bus and the optical cache controller. The OE cache controller is implemented in a Xilinx 4K series FPGA. It accepts address and synchronization signals from the PCI controller and uses 4-way set associative mapping to determine if an address is available in the optical cache. If it is, the SRAM modules in the cache are accessed and data is returned directly to the PCI bus via the Bus Mux/Ecc circuitry. If not, the address of the requested page is passed on to an external optical memory controller which initiates an optical memory access. Once the requested page of optical memory is available at the detectors, incoming data is transferred in parallel by line from the detector array into the OE cache SRAMS. Synchronization and control of the transfer is handled by the OE cache controller, and once again, the incoming data is routed via the Bus Mux/Ecc circuitry. Simultaneously with loading the incoming optical data, the cache controller monitors the data stream and latches the requested word into an internal buffer. This allows the memory request to be satisfied without the additional cycles required for a second access to the OE cache SRAMS.

Cache organization

In this implementation the SRAMs in the OE cache are accessible either as 32-bit words on the PCI bus or as 512-bit rows from the optical memory page. Figure 4 shows how this organization is mapped into the memory array by the Mux/Ecc modules. Each cell in the figure is a 16-bit word and the entire array appears to the OE cache controller as a 4Mb linear address space. These cache addresses are shown to the left of the diagram. Each PCI bus reference accesses up to 4 bytes of memory, such as in the block shown diagonally shaded in the figure. The dark shaded column in the figure represents all of the data in a

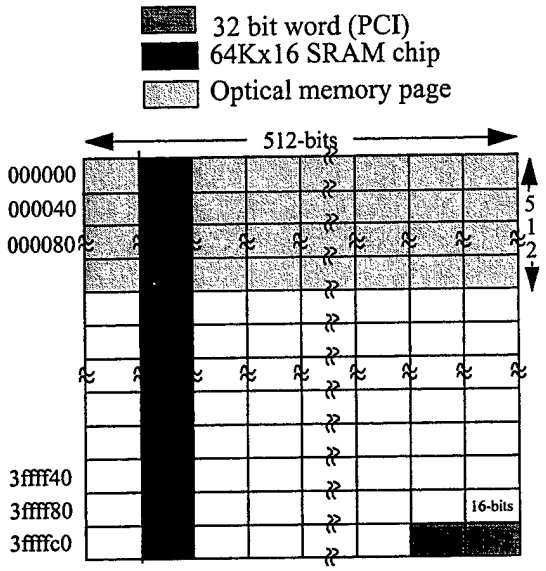


Figure 4: Cache memory addressing

single 64Kx16 SRAM chip. Thirty two such SRAM chips are accessed in parallel from the optical memory interface and each page of optical memory is loaded into 512 sequential "row" addresses as shown by the medium grey shaded area. Thus, each page of optical memory is mapped into the cache address space as a 32KB line of OE cache.

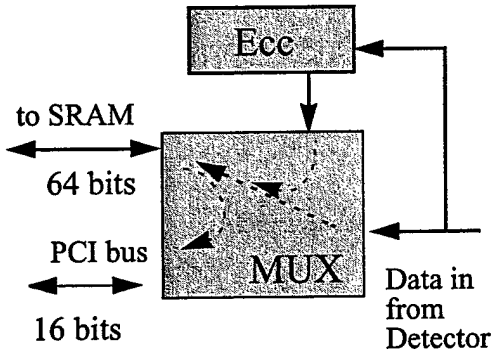


Figure 5: Bus Mux/Ecc configurations

Figure 5 shows a diagram of the contents of the Mux/Ecc chips. Also implemented in Xilinx 4K series FPGAs, the role of the these chips is two-fold. First, they support the data paths necessary for alternately accessing the memory by optical page line and PCI word. As shown by the dotted lines in the figure, data paths can be established in each chip between 64 bits of an incoming optical memory line and the SRAM I/O pins. In another configuration any 16 of the 64 SRAM I/O lines can be selected for transfer to the PCI bus. Second, these chips provide for the introduction of Error checking and correcting (Ecc) logic operating over the incoming data stream. The reconfigurable logic devices as well as the position of the chips relative to the data streams allows experimentation with a variety of algorithms operating on the data either as it is incoming from memory, or outgoing to the PCI bus.

Performance analysis

Since the prototype system contains both an electronic and an optoelectronic memory hierarchy we have constructed a simulation model capable of analyzing the relative performance of a program running either with its address space defined on the swapping disk, or in the optical memory. In both cases, the program uses the same primary and secondary cache structure. For the primary memory/swapping disk case, secondary cache misses are handled with a fixed access time of 100ns for hits. Secondary cache misses in the OE memory case are handled with a fixed access time of 150ns for hits. The additional access time is due to the additional overhead of memory access via the PCI bus versus the processors private memory bus. The model for access to the swapping disk assumes an average seek time of 10ms and a data transfer rate of 100Mb/s. Page size for the primary memory is 2K

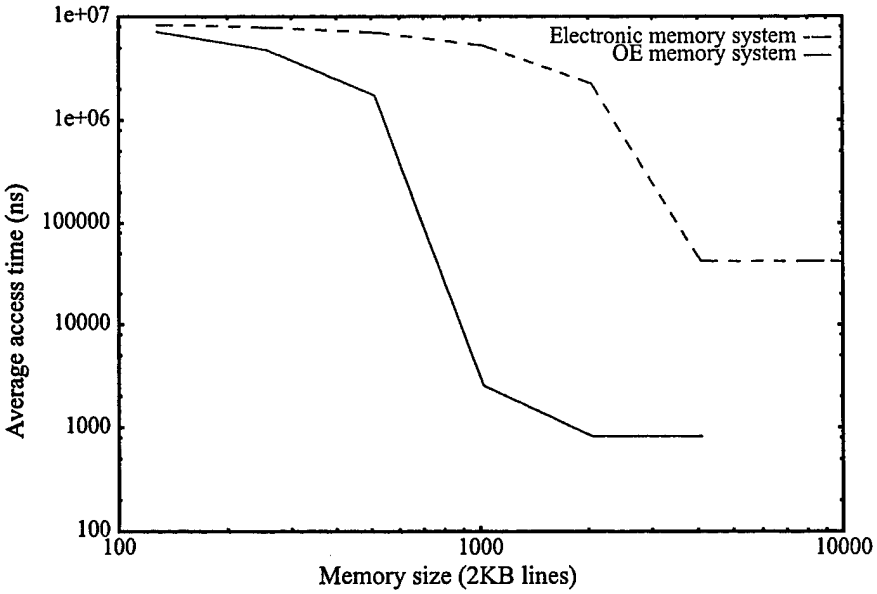


Figure 6: Access time vs. Memory/Cache size for OE and Electronic memory system (logarithmic scales)

bytes. The optical memory is assumed to have an average access time of 10ms and a page size of 512x512bits, or 32K bytes. Both models use NUR replacement to select the locations of incoming blocks.

The benchmark program was an implementation in C of the heap sort algorithm with the code instrumented to trace all memory accesses. The fault rate was recorded as the program was run under each of the two memory models while varying the size of the primary memory or OE cache for each run. Figure 6 shows a plot of the average memory access time versus memory size for each of the memory models.

This plot demonstrates the performance improvement of the OE versus electronic memory system. As the memory (or cache) size is increased, the hit rate, (the number of references accessed from within the cache) increases and the average memory access time decreases. In the case of the OE cache, the large block size causes the hit rate to increase more quickly as more data is brought in per transfer. Since the average access time is a weighted sum of the access time of each level, weighted by the hit ratio, the larger block transfer has a significant impact on the average access time even for relatively high hit ratios. Although both systems have approximately the same latency for a miss, OE cache misses transfer sixteen times more data than a corresponding miss in the electronic system. Therefore, the OE cache pays the latency penalty for misses far less often that an electronic implementation with the same memory size. Even in the limit, as the memory size grows to accommodate the application, the parallel transfer rate of the OE memory system shows a two order of magnitude difference in performance.

Future Work

We intend to use the prototype to investigate a number of performance related issues as well as algorithms for 3D spatial Ecc. Additionally, we are currently investigating the relationship between various competing technologies for the optical memory and the smart pixel array used in the cache. These technology choices must be considered in the context of architectural issues such as the address translation mechanism, frame size at each level of the memory hierarchy, write policy, replacement algorithms, and coherency support mechanisms for multiprocessor implementations.

This work has been supported in part by the United States Air Force, AFMC/Rome Laboratory/PKRL, under contract #I-6-4284.

References

[1] D.M. Chiarulli and S.P. Levitan, "An Optoelectronic Cache Memory System Architecture", Applied Optics Vol. 35, No. 14,pp. 2449-2456, 10 May 1996

[2] P. A. S. Dvornikov, S. Esener, and P. Rentzepis,"Three-dimensional optical storage by means of photon interactions,"Optical Computing Hardware, ed. Jahn and Lee, Academic Press 1994.

An Optoelectronic Cache Memory System Architecture

Donald M. Chiarulli
Department of Computer Science

Steven P. Levitan
Department of Electrical Engineering

University of Pittsburgh
Pittsburgh, PA 15260

ABSTRACT

We present an investigation of the architecture of an *optoelectronic cache* which can integrate terabit optical memories with the electronic caches associated with high performance uni- and multi- processors. The use of optoelectronic cache memories will enable these terabit technologies to *transparently* provide low latency secondary memory with frame sizes comparable to disk-pages but with latencies approaching those of electronic secondary cache memories. This will enable the implementation of terabit memories with effective access times comparable to the cycle times of current microprocessors. The cache design is based on the use of a smart-pixel array and combines parallel free space optical I/O to-and-from optical memory with conventional electronic communication to the processor caches. This cache, and the optical memory system to which it will interface, provides for a large random access memory space which has lower overall latency than that of magnetic disks and disk arrays. In addition, as a consequence of the high bandwidth parallel I/O capabilities of optical memories, fault service times for the optoelectronic cache are substantially less than currently achievable with any rotational media.

This research has been supported, in part, by a grant from the Air Force Office of Scientific Research under contract number F49620-93-1-0023DEF

Introduction

Hierarchical memory architectures are based on two fundamental paradigms of computer architecture: a hardware paradigm that smaller is faster, and a software paradigm that programs access memory in patterns that exhibit spatial and temporal locality. Thus the latency inherent in the access to a large memory can be hidden in a pyramid with small and fast memory modules at the top, closest to the processor, and large, slower, memories at the bottom. Optical and optoelectronic memory devices offer the potential for building very large memories at the lowest level of the hierarchy. Unlike magnetic disks, optical memory provides random access throughout the address space as well as high bandwidth and highly parallel data transfers. Key to the successful design of such a system is the resolution of architectural issues such as the address translation mechanism, frame size at each level, write policy, replacement algorithms, and coherency support mechanism[PHH88]. Recent developments in the integration of silicon and optoelectronic technology such as FET-SEEDs or VCSELs [OKKea95, Nef94, Dic91, LM93, ea95] have provided the devices necessary to build such a system. In this paper we present an investigation into the feasibility and design parameters of a seamless optoelectronic memory hierarchy which utilizes an optoelectronic cache.

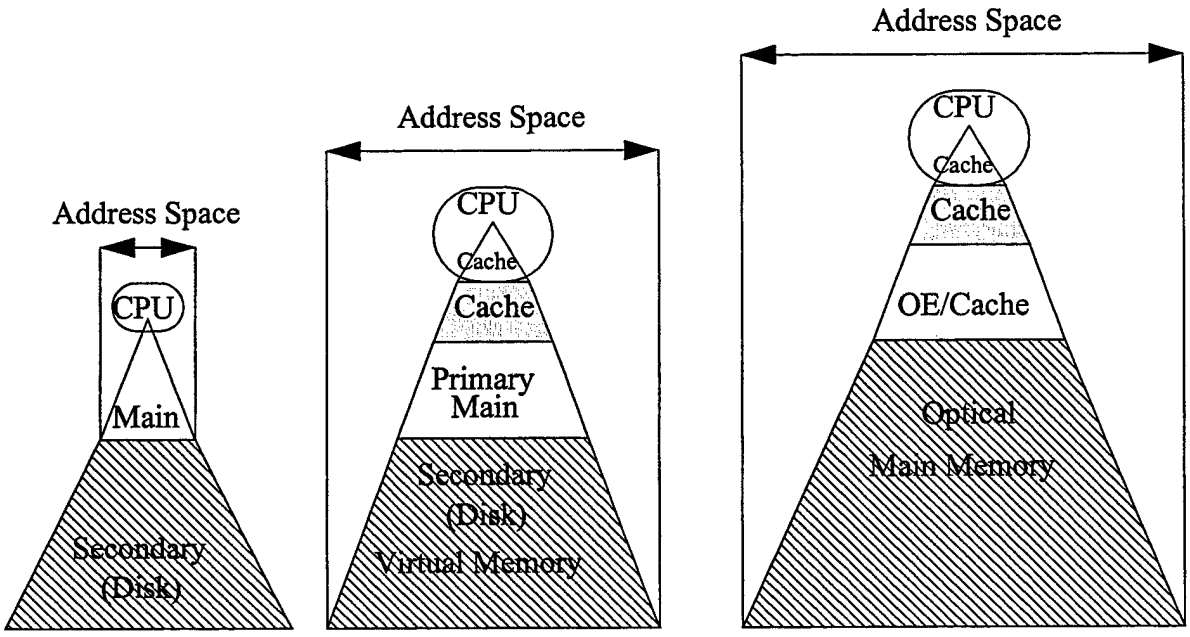


Figure 1: Memory Hierarchy Evolution

As shown in Figure 1, in the conventional description of a memory hierarchy, a distinction is made between secondary memory, primary memory, and each level of cache memory. This distinction was originally based on the *visibility* of the memory relative to a machine language instruction. In this historical context, shown in Figure 1(a), primary, or main, memory was defined by the program address space and secondary memory, or backing store, was associated with input and output. Caches, to the extent they existed, were invisible, and were first implemented as a buffer between the processor and primary memory. In modern systems, shown in Figure 1(b), caches are implemented routinely and typically exist in multiple levels, with the first

level cache integrated into the processor itself. The distinction between primary and secondary memory has been significantly blurred by address segmentation and virtual memory systems. Typically, secondary memory now supports the program address space, parts of which are swapped on demand into a semiconductor RAM primary memory level. In this proposal, we dispense with the notion of distinct primary and secondary memories. As shown in Figure 1(c), we merge these levels into a single optoelectronic memory at the lowest level of the hierarchy. The processor address space is directly supported in the optical memory. All levels between this optical memory and the processor are transparent to the processor and therefore are referred to as cache levels.

Figure 2(a), shows a block diagram of a physical realization of an optoelectronic memory system for a uniprocessor and Figure 2(b) shows a realization for a multiprocessor application. Reflected in these designs is the fact that most state-of-the-art processors use a two level cache at the top of the memory hierarchy with a cache controller for these levels integrated on the processor chip. The top level, or primary cache is a small on-chip memory. The secondary cache is somewhat larger and is off chip. These caches typically have access times on the order of the processor clock period and data transfers between them are in the range of ten to one hundred words. At the lowest level, the optical memory provides high capacity data storage. Between these levels, the *optoelectronic cache* level links the secondary cache with the optical memory level. The optoelectronic cache is a dual ported electronic memory with an optical port which connects to the optical memory and an electronic port which connects to the levels above.

In the shared memory multiprocessor application shown in Figure 2(b), the optoelectronic cache serves the same function. However in this design, it is assumed that the cache is either multiported or banked to provide multiple access points on the electronic interface. An interconnection network exists between the optoelectronic cache and the processor secondary caches. Alternative designs may eliminate this interconnection network by duplicating the optoelectronic

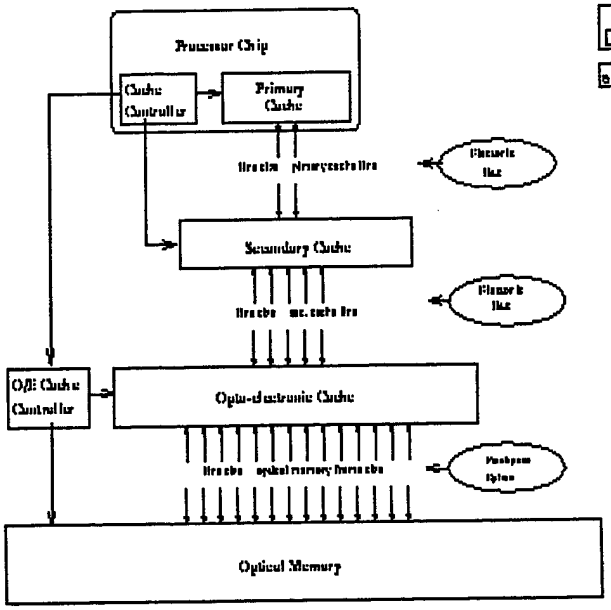


Figure 2(a): Uniprocessor Optoelectronic Memory Hierarchy

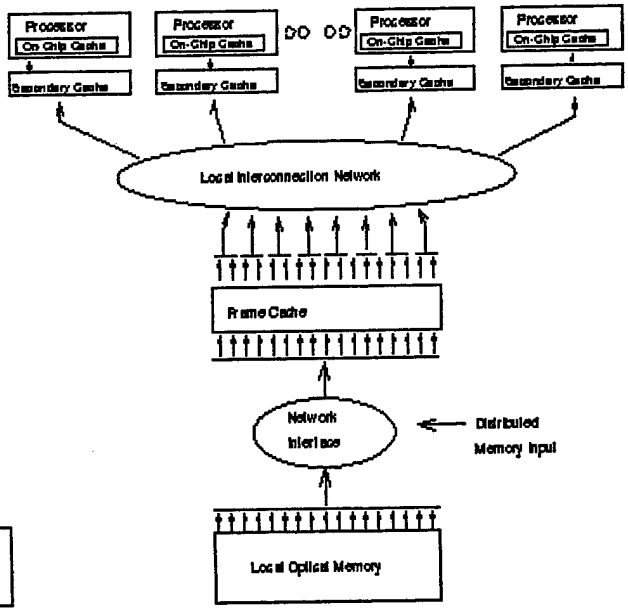


Figure 2(b): Multiprocessor Optoelectronic Memory Hierarchy

cache at each processor and providing interconnect at the optical memory level. Similarly, both local and distributed memory models might be supported since it is possible to implement the optical memory such that both local and networked banks of memory exist. Thus, the optoelectronic cache is also an enabling technology for future multiprocessors that use large shared optical memory systems.

In this paper we present an investigation into the design of a new optoelectronic cache level that will interface a terabit optical memory to the electronic caches associated with one or more processors. The cache level is based on the use of smart-pixel array technology and will combine parallel free-space optical I/O to an optical memory with conventional electronic communication to the processor caches. The optoelectronic cache, and the optical memory system to which it will interface, will provide for a large random access memory space that will have a lower overall latency than magnetic disks or disk arrays.

In the next section we briefly present the context of current, or proposed optical memory systems and present our model for the optoelectronic interface to these memories. Next we outline a specific design for an optoelectronic cache and cache controller. We then present a preliminary performance analysis based on analytical results and simulation data. We conclude with our anticipated results and a workplan to achieve our research goals.

Background

There are a number of competing optical memory technologies currently being investigated. We focus on non-rotational read/write media. This is because the *access time* of all rotational media based systems precludes their use as operating system transparent memories. That is, the latency of these devices would necessitate a process context switch in the case of a fault.

"3-D" optical memory systems on the other hand, have the potential of both fast access times as well as large capacities [Ese89]. Typical examples of these systems are:

- Spectral hole burning for memories at low temperatures [WBB85, CRW91], and the possibility of room temperature devices [ALWR91].
- Photorefractive materials for holographic storage [BMP94].
- Two photon systems [FHPea93].

All have the common characteristics of high access bandwidth, supported largely by parallel access. Specifically, each reference returns a *frame* of data, where the term frame refers a large collection of bits typically related by membership in a specific data structure such as an image bitmap. In this discussion we select a less restrictive and technology independent model for the optical memory. The model assumes only that it is a high capacity memory with access parallelism modelled as a long word length. As with a conventional memory hierarchy, the access time is assumed to be significantly longer (two to three orders of magnitude) than the clock period of the processors. Input and output ports for the optical memory are assumed to be a free space optical interconnect with the number of channels corresponding to the number of bits in the memory word. However, given current or near term technology limits, it may be necessary to multiplex the optical system in order to accommodate limitations on the density of EO device integration.

Figure 4: OE cache address consisting of optical memory word and offset

Optoelectronic Cache Architecture

In this section we present a realization of the optoelectronic (OE) cache level in the OE memory hierarchy. As shown in Figure 1(c), the cache is in the same position as the primary memory in a conventional hierarchy. However, unlike primary memory it is transparent to both the processor and the operating system. This level is the interface between the optical memory backing store, and the secondary cache associated with the processor. Another distinguishing feature of the OE cache is its significantly larger *line size* then is typical for primary memory. A *memory line*, (also commonly known as a cache line) is the amount of data transferred between levels of the hierarchy when a memory fault (or equivalently, a cache miss) occurs. Thus, the size of a line at a particular level, is a trade-off between the locality supported within the memory traffic and the efficiency to which the cache is utilized. A large cache line more loosely constrains memory access locality. However, large cache lines will also bring into the cache fragments of unused memory. This effect is called *internal fragmentation.* In a conventional memory the cost associated with internal fragmentation can be significant since the fault service time is typically linearly related to the line size. However, in the OE cache, the (much larger) line size is determined by the width of the optical memory word. The parallel access characteristics of an optical memory make it possible to transfer cache lines to and from the optical memory in a single access time. This is substantially faster than the equivalent transfer from a magnetic disk which must allow for both rotational latency and serial transfers. This is a significant advantage. This obviously has an effect on the organization of the cache itself, and also impacts the mechanism for address translation and, in multiprocessor systems, coherency issues.

Figure 3 shows a block diagram of a design for the OE cache. In this diagram, optical I/O is transmitted and received by an array of SEED devices shown on the right. The electronic bus, drawn vertically on the left, connects the OE cache to the electronic secondary cache level. The cache itself is modelled as a two dimensional array of bits. Each column holds one cache line which corresponds to one word (frame) from the optical memory. Each column is subdivided into words, each the width of the processor memory bus. Each of these words is in turn connected to the electronic I/O bus.

When a fault occurs in the secondary cache, the optoelectronic cache controller processes the address to determine if there is a cache hit in the OE cache. In other words, if the requested location is present in the OE cache. If a hit occurs, the controller translates the address of the requested location from its location in the processor address space, to an address within the OE cache. This address is partitioned as shown in Figure 4. Once translated, a pair of decoders handle the cache address. One decoder reads the high order address bits and selects all of the bits in a single column. Another decodes the low order bits and selects one electronic memory word within the selected column. Thus, when the memory is accessed from the optical memory side, an entire cache line is read or written simultaneously. While on the electronic side, a single word is selected by enabling both the corresponding cache line (column) and the corresponding word offset onto the electronic bus.

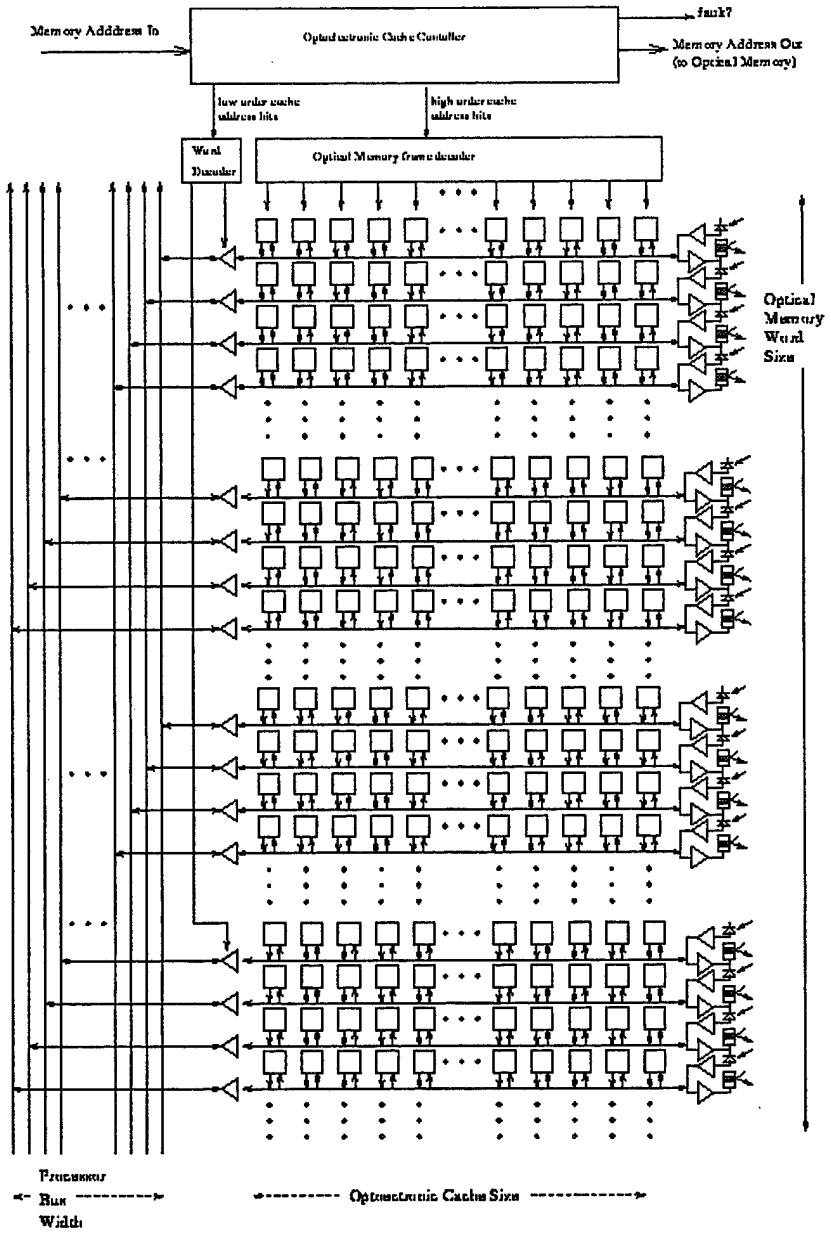


Figure 3: Optoelectronic cache block diagram

We assume that, although Figure 3 shows the optoelectronic cache as a monolithic implementation, it is likely that an actual implementation may partition the memory both along the word width and memory depth. Also, given the relative bandwidth of the optical interconnect to the two memories, it is possible that the optical system may be multiplexed in order to reduce the device count. Error detection and/or correction mechanisms can also be built into the optoelectronic cache by connecting this hardware to the horizontal bus carrying the optical memory word.

Controller Architecture

The discussion of address decoding in the previous section assumed a cache hit. In other words, the cache controller shown at the top of Figure 3 interpreted an incoming memory address, determined that the desired location was in the cache, and translated the memory address into a cache address. In this section we will briefly describe how such a translation might take place in the optoelectronic cache controller.

Consider an *n-bit* memory address corresponding to a location in the data memory space of a processor. Assuming that the word size in the processor memory space and the word size of the optical memory are powers of two, this location is at some specific offset within a larger optical memory word. Thus the n-bit address is partitioned into fields corresponding to a location in optical memory and the offset of the word. This organization is identical to the one shown in Figure 4 with the exception that the high order bits now identify an optical memory word within the address space of the processor. In fact, it is identical to the organization of addresses at any level of the memory hierarchy where the high order bits select a specific memory line and the low order bits select the offset within the line.

The number of bits in the high order partition of these addresses is determined by the relative sizes of the memory address space and each of the cache levels. Address translation is the operation of mapping from a value for the high order bits in the memory address space to the high order bits (cache line number) of a cache address. There are a number of methods for accomplishing this translation which are well documented in the literature on memory systems [Sta87]. They include low latency solutions which use direct and fixed mappings. More complex methods use associative memory lookups, and others use hierarchical tables. Each has different characteristics for latency, implementation efficiency, and cache utilization. In general, address translations mechanisms with higher latencies tend to use the cache more efficiently and tend to support lower the fault rates. Thus, if the cost of fault is high, such as the case for swapping to and from disk, then a designer is willing to tolerate a higher latency in address translation (for either a hit or a miss) in order to minimize the frequency of faults. For example, in a conventional memory hierarchy between primary memory and a swapping disks, fault costs can typically be on the order of milliseconds. Hence, the dynamic address translation algorithms used in a virtual memory system may add latency of two or three times the normal memory latency as overhead, in order to implement nearly optimal replacement strategies. With an optical memory at the lowest level of the hierarchy, these fault costs are reduced to microseconds. Thus a significantly faster (but less optimal) address translation mechanism can be successfully implemented.

Throughout this discussion we have assumed that the optical memory replaces both the primary memory and disk backing store of a conventional memory system. Thus the traditional notion of a "virtual memory" as a process level address space is replaced by a single, large, processor level address space. This design is consistent with the current trends in processor design in which 64-bit address spaces are quite common. When an optical memory technology is used to populate these huge address spaces, conventional mechanisms for memory management in operating systems will be obsolete. Both virtual memory mechanisms as well as file system organizations will be replaced by common name space object oriented operating systems [TM93][Ant90]. In the near term, however, it is still possible to integrate the proposed optical memory system into

conventional virtual memory operating systems, which assume a unique address space for each process, by simply making an association between the upper bits of a optical memory address with the process-id of a running process.

Performance Analysis

In this section, we present an analysis of the relative performance of the OE memory system architecture in comparison to traditional memory hierarchies.

The average memory latency $\overline{L}_x$ at any level, x, of a memory hierarchy can be calculated as:

$$\overline{L_x} = (1 - p_x)L_x + p_x L_{(x-1)}$$
$$L_0 = L_{BackingStore}$$

where p_x the fault probability, (1 - p_x) is the hit probability, and L_x is the memory access time at level x. L_0 is the latency associated with the memory at the lowest level of the hierarchy, commonly known as the backing store. In this expression we approximate the miss penalty at all but the lowest level to the average latency of the next lower level. This approximation is accurate if we assume that memory banking, or other prefetching techniques have been implemented between these levels. At the $L_{0,}$ level, specifically when disk drives are used as the backing store, is it necessary to consider the transfer time of a memory line as part of the latency. In this case, if T_s is the average seek time, T_r, is the average rotational latency and T_x is the transfer rate of a disk based backing store, the miss latency of a memory line of size n_m is:

$$L_0^{disk} = (T_s + T_r + n_m T_x)$$

Alternatively, when an optical memory is used as the backing store and the entire cache line is transferred in parallel, only T_o, the access time of the optical memory, needs to be considered:

$$L_0^{optical} = T_o$$

Taking only this difference into account, Figure 5, is a plot of the average latency versus the hit rate for two, single level, memory systems. One uses disk technology as the backing store, the other uses an optical memory as the backing store. For this plot, L_1 is set to $10ns$, T_o is set to $1us$ and the average disk latency is assumed to sum to 1ms. Latency on the y axis is plotted on a log scale and hit rates are varied only in the range of 90 to 100 percent. Clearly, the large region between the lines represents the potential latency improvement with an optical backing store. However, this improvement assumes that a given application will fault at same rate in both systems. This may not be true, since, in the optoelectronic memory system, the line size must be significantly larger in order to exploit the parallelism in the memory transfers; while the corresponding line size in the electronic/disk based memory system will tend to be smaller, primarily in order to reduce the transfer component of the miss latency.

The factors which influence the choice of line size in a memory system are the locality behavior of the applications, the acceptable level of internal fragmentation, the size and complexity of the tables used by the memory controller and the miss penalty associated with loading the memory line into the cache. For hierarchies based on disk and disk arrays as the backing store, primary memory line sizes (pages) are typically on the order of 128 to 4096 bytes. Assuming an optical memory frame size of 1Mb, the corresponding line size for an optoelectonic cache is 128k bytes.

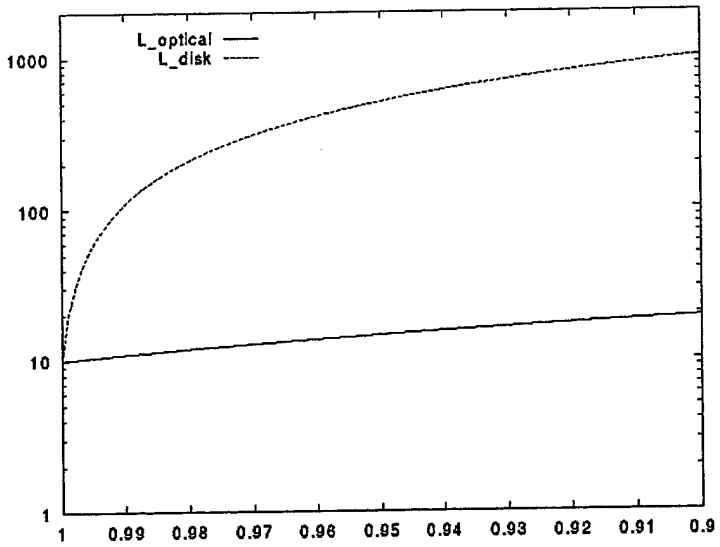


Figure 5: Latency (ns) vs. Hit Rate (%)

Clearly, the optoelectronic memory hierarchy has the ability to replace a much large cache line with a lower penalty. However, given that such a large cache line will have a correspondingly larger amount of internal fragmentation we can expect a corresponding increase in the fault rate. In the next section we show that, for several applications, these internal fragmentation effects are minor in comparison to the performance gained due to reductions in the miss penalty.

Simulation Results

To investigate the relative fault rates of an optoelectronic memory hierarchy verses a conventional electronic/magnetic disk based hierarchy, we implemented models of two memory systems. Each has a three level hierarchy. The first version models the behavior of an electronic primary memory at level one with magnetic disk as the backing store. The second version models the behavior of an optoelectronic cache at level one with optical memory as the backing store. The top two levels in both models are electronic primary and secondary cache memories with identical characteristics. The sizes and latency associated with each level are summarized in Table 1.

Cache Level	Size	Lines	Line Size	Hit Latency	Miss Penalty
primary	512Kb	4096	64 bytes	10 ns	(lower level hit)
secondary	2 Mb	4096	256 bytes	50 ns	(lower level hit)
opto-cache/ main memory	variable	variable	128Kb/ 2048 bytes	100ns	(lower level hit)
optical memory				1000 ns	
main / disk				Seek +Transfer (avg=1ms)	

Table 1: Simulation Parameters

Address translation at each level was modelled using direct mapping in the primary cache, set associative mapping in the secondary cache and dynamic address translation (page tables) in the main memory/optoelectronic cache. Direct mapping in the primary cache was implemented as follows. For each address, the low order six bits were treated as the offset within the 64 byte cache line. Of the remaining bits, which constituted the line number, the low order 12 bits were used to select one of the 4096 cache lines in the primary cache. The remaining high order bits in the memory line were compared to a tag field attached to the selected cache line and which held the high order bits of the memory line currently stored in the cache. A match with the requested line constituted cache hit. A miss required access to the secondary cache. Address translation in the secondary cache was set associative. Like direct mapping, the low order six bits, the offset portion, of the address were removed. The remaining bits were partitioned such that the low order 10 bits were used to select one of 1024 sets of 4 cache lines. In this case a cache hit occurred if the tag field of any of the four cache lines in the selected set matched the requested cache line. In the case of the main memory/optoelectronic cache level, table driven dynamic address translation was modeled. In an actual implementation this would mean that the line number portion of the address would be used to index a table that contained the address of the corresponding cache line, in the case of a hit, or a flag in the case of a miss. In order to save memory in the simulator this was implemented as a linear search of the tag fields in the cache (with no time penalty). This model is also functionally equivalent to a full associative address translation.

Based on these two models, three applications, an image convolution, a heap sort, and a matrix multiplication were coded in C and instrumented to provide memory address traces for all data memory read operations. Each application was sized to run in a 4Mb address space. Two sets of runs were made for each application, one set for each memory model. Each set of runs con-

sisted of running the application in successfully smaller total memory sizes. Sizes ranged from 64Mb to 128Kb (256Kb for the optical runs) for each set. The first set assumed a line (page) size of 2048 bytes and a fault latency of 1 millisecond for seek + DMA transfer time of (100 × 2048) ns. The second set assumed a 128Kb page size and 1000ns miss penalty. Figure 6 is a plot of the hit rate versus memory size, primary memory (Ro) or optoelectronic cache (Oo), at level one.

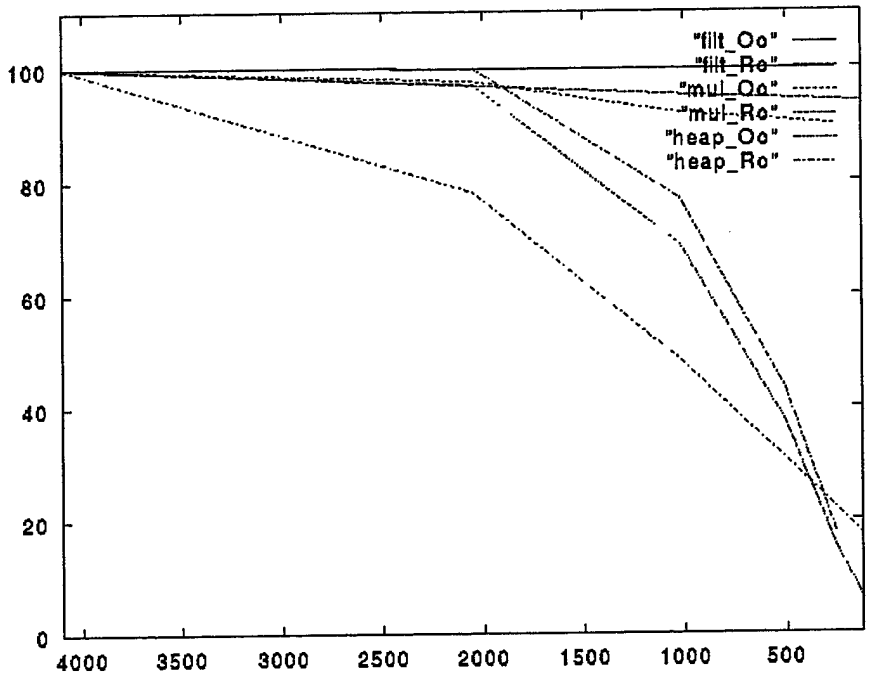


Figure 6: Hit Rate (%) vs. Memory Size (Kb)

The data in Figure 6 is plotted for memory sizes ranging from 4Mb, the size at which the entire application can be loaded, down to 128Kb, which corresponds to a single line in the optoelectronic cache. The results demonstrate, for these applications, that internal fragmentation effects do not decrease the hit rate (increase the fault rate) except in the case of the heap sort application with a very small optoelectronic cache (less than 512Kb). In all other cases, the increases in fault rate due to internal fragmentation is entirely offset by a reduction in faults caused by the greater amount of data transferred per fault.

Returning to the latency calculations, we can now determine how these effects combine and compare average latency of the memory systems as a whole. The average latency of the electronic/disk memory system can be recursive constructed as follows:

$$\overline{L_{disk}} = (1-p_p)L_p + p_p((1-p_s)L_s + p_s((1-p_m)L_m + p_m(T_s + T_r + n_m T_x)))$$

where L_p, L_s, *and* L_m are respectively, the access latency of the primary cache, secondary cache, and primary memory, and p_p, p_s, and p_m are the fault rates for the primary cache, secondary cache, and primary memory. Similarly, for the optoelectronic memory system, the average latency can

be written as:

$$\bar{L}_{optoelectronic} = (1-p_p)L_p + p_p((1-p_s)L_s + p_s((1-p_{oc})L_{oc} + p_{oc}T_o))$$

where p_{oc}, and L_{oc}, are the hit probability, and latency of the optoelectronic cache. Using the specifications in Table 1 and the fault rate data from the simulations, the average latency of each memory system is computed and plotted in Figure 7 for the three applications tested. The results are plotted on a log scale for memory size and latency. The latency plotted is the average memory access time though all levels of the memory hierarchy.

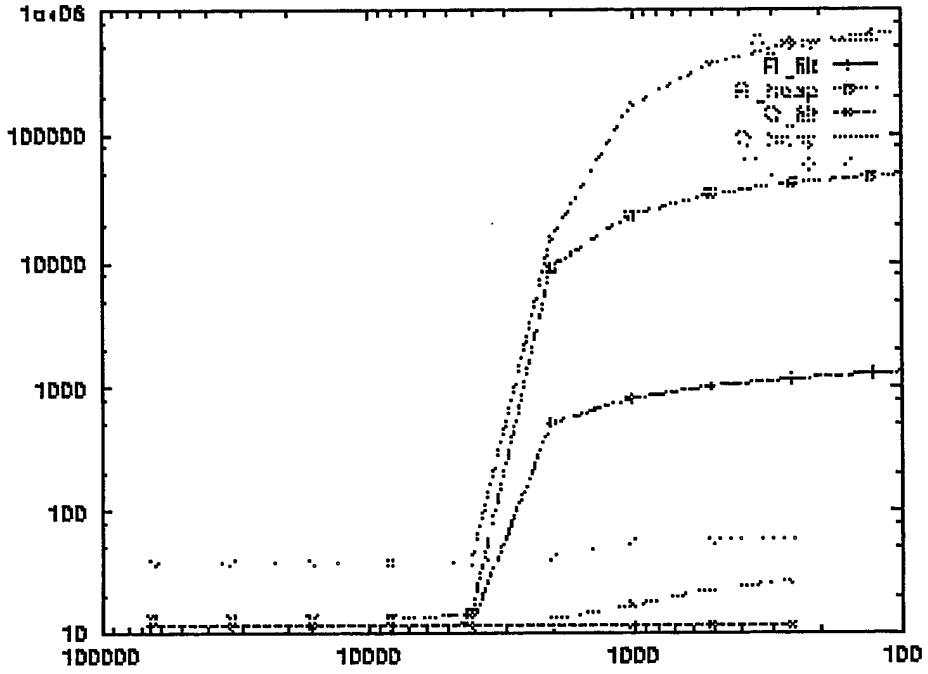


Figure 7: Latency (ns) vs. Memory Size (k)

Discussion

For the applications tested, the simulations show a three to four order of magnitude increase in the performance of the optoelectronic memory system versus that of a conventional memory hierarchy. Thus, we have demonstrated that optoelectronic cache memories can be used to effectively interface a low latency optical backing store for an optoelectronic memory hierarchy. Although line sizes in the cache are typically larger than disk-pages, average memory access latency is not adversely affected by the additional internal fragmentation introduced. We are currently investigating the relationship between various competing technologies for the optical memory and the smart pixel array used in the cache. These technology choices must be considered in the context of architectural issues such as the address translation mechanism, frame size at each level of the memory hierarchy, write policy, replacement algorithms, and coherency support mechanisms for multiprocessor implementations.

References

[ALWR91] S. Arnold, C. T. Liu, W. B. Whitten, and J. M. Ramsey. Room-temperature microparticle-based persistent spectral hole burning memory. *Optics Letters*, 16(6):420–422, 1991.

[Ant90] V. G. Antonov. A regular architecture for operating systems. *Operating Systems Review*, 24(3):22–39, 1990.

[BMP94] G.W. Burr, F. H. Mok, and D. Psaltis. Storage of 10000 holograms in LiNbO3:Fe. In *1994 Technical Digest Series: Proceedings of 1994 Conference on Lasers and Electro-Optics and The International Electronics Conference CLEO/IQEC*, volume 8, page 9, Washington DC, May 1994. Optical Society of America.

[CRW91] C. De Caro, A. Renn, and U. P. Wild. Hole burning, stark effect, and data storage: Holographic recording and detection of spectral holes. *Applied Optics*, 30(20):2890–2898, July 1991.

[Dic91] A. Dickinson. An optical respite from the von neuman bottleneck. In *Technical Digest: Spring Topical Meeting on Optical Computing*, volume TuC4-1, pages 239–242. Optical Society of America, 1991.

[ea95] A. V. Krishnamoorthy et. al. Implementation of a photonic page buffer based on GaAs MQW modulators bonded directory over active silicon VLSI circuits. In *Postdeadline Paper, Spring Topical Meeting on Optical Computing*. OSA, 1995.

[Ese89] S. Esener. 3-D optical memories for high performance computing. Spatial light modulators and applications III. In *SPIE Critical Review of Technology Series*, volume 1150, pages 113–119. SPIE, 1989.

[FHPea93] J. E. Ford, S. Hunter, R. Piyaket, and Y. Fainman et al. 3-D two photon memory materials and systems. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 1853, pages 5–13. SPIE, 1993.

[LM93] A. L. Lentine and D. A. B. Miller. Evolution of the SEED technology: bistable logic gates to optoelectronic smart pixels. *IEEE Journal of Quantum Electronics*,

29(2):655–669, February 1993.

[Nef94] J. Neff. Optical interconnects based on two-dimensional VCSEL arrays. In *Proceedings of the First International Workshop on Massively Parallel Processing Using Optical Interconnections*, pages 202–212. IEEE Comput. Soc. Press, 1994.

[OKKea95] I Ogura, K. Kurihara, S. Kawai, and M. Kajita et. al. A multiple wavelength vertical-cavity surface-emitting laser (VCSEL) array for optical interconnection. *IEICE Transactions on Electronics*, E78-C(1):22–27, 1995.

[PHH88] S. Przybylski, M. Horowitz, and J. Hennessy. Performance tradeoffs in cache design. In *Proceedings of the 15th Annual Symposium on Computer Architecture*, volume 16(2), 1988.

[Sta87] William Stallings. *Computer Organization and Architecture*. MacMillan, 1987.

[TM93] R. Trehan and K. Maeda. A distributed object oriented language and operating system. In *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences*, volume 2, pages 70–79. IEEE, 1993.

[WBB85] P. Wild, S. E. Bucher, and F. A. Burkhalter. Hole burning, stark effect, and data storage. *Applied Optics*, 24(10):1526–1530, 1985.